Universidad Nacional Autónoma de México

Facultad de Ingeniería

Laboratorio de Computación Grafica e Interacción Humano-Computadora

Manual técnico - Proyecto Final

Profesor Ing. Edén Espinoza Ursúa

Integrantes del equipo:

Gonzalez Franco Emilio

Peláez Semprun Diego Eduardo

Fecha de entrega: 09 de mayo de 2025

Semestre 2025-2

Objetivos

- Recrear en 3D la fachada y un espacio interior del universo de la serie animada “El Increíble Mundo de Gumball”.
- Representar fielmente al menos siete objetos escogidos del entorno con texturas, iluminación y materiales aplicados.
- Construir un entorno interactivo navegable en OpenGL.

Diagrama de flujo del software

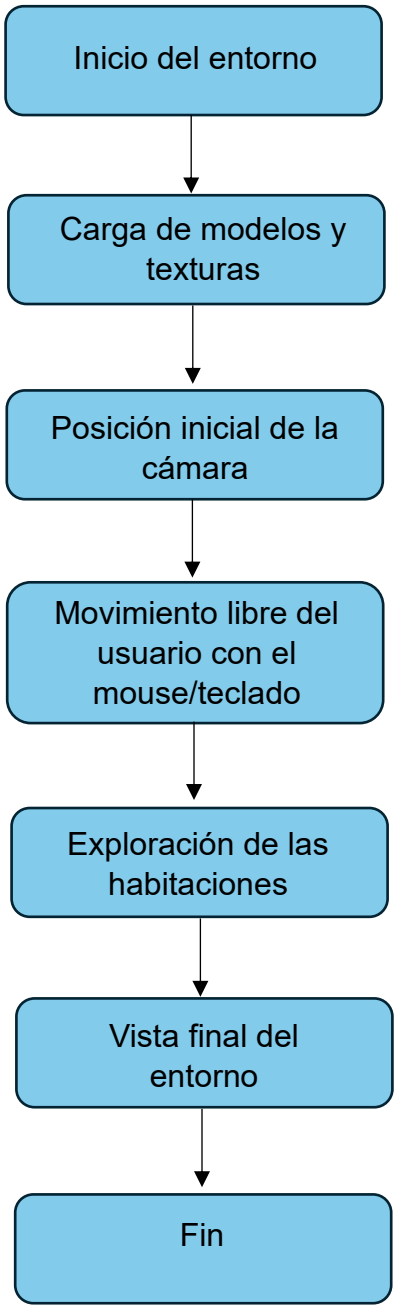
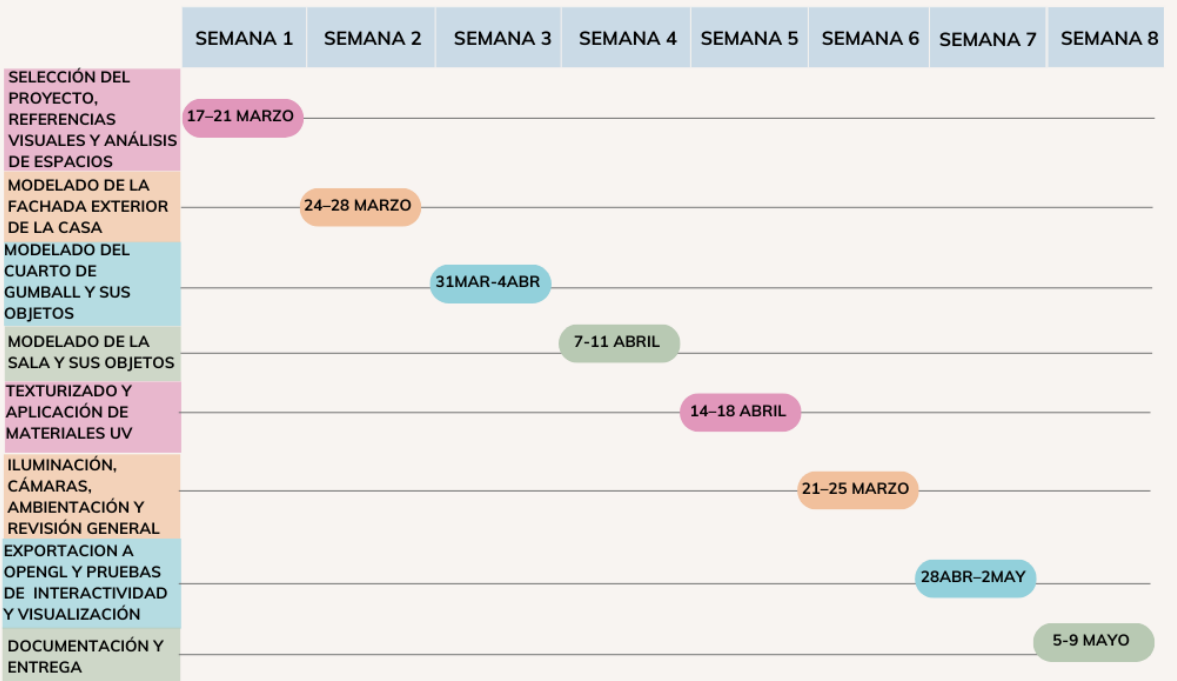


Diagrama de Gantt



Alcance del proyecto

- ✓ Se modeló en Blender tanto la fachada de la casa de Gumball como el cuarto, la sala y los objetos de cada uno.
- ✓ Los objetos modelados incluyeron el sofá, el televisor junto con su mueble, la lámpara, la mesita de centro, el escritorio, la computadora, la litera, el baúl, el balón de futbol americano, entre otros.
- ✓ Se aplicaron texturas personalizadas lo más parecidas o fieles posibles mediante UV mapping.
- ✓ El entorno permite exploración visual con una cámara sintética libre tanto en Blender como en OpenGL.

Limitantes

- El entorno no incluye interacción con animaciones ni físicas debido a restricciones de tiempo y ajenas.
- Se priorizó la fidelidad visual lo cual puede ser tanto bueno como malo.
- Algunos detalles de la serie original como por ejemplo la forma del cuarto con respecto a la forma de la casa, aunque no tiene mucha lógica, se adaptó fielmente.
- El uso de transparencia en las ventanas no fue posible debido a la compatibilidad de Blender con OpenGL

Metodología de software aplicada

Se empleó un enfoque basado en etapas del modelo cascada, siguiendo estos pasos:

1. Investigación y análisis de referencias
2. Modelado progresivo de escenarios y objetos
3. Aplicación de materiales y texturas
4. Ambientación con luces y cámaras
5. Pruebas de visualización e iteración
6. Documentación técnica y de usuario

Documentación del código

Lo primero que se hace en el código es incluir las librerías que nos permiten representar el entorno en tres dimensiones.

```
// ================================
// INCLUDES
// ================================
#include <iostream>
#include <cmath>
#include <GL/glew.h>
#include <GLFW/glfw3.h>
#include "stb_image.h"
#include "SOIL2/SOIL2.h"
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>
#include "Shader.h"
#include "Camera.h"
#include "Model.h"
```

Posteriormente se declaran las funciones que controlan las acciones del programa al momento de recibir una entrada por parte del usuario.

```
// ============================
// FUNCTION DECLARATIONS
// ============================
void KeyCallback(GLFWwindow* window, int key, int scancode, int action, int mode);
void MouseCallback(GLFWwindow* window, double xPos, double yPos);
void DoMovement();
```

Luego definimos las variables globales y las constantes, que definen cosas como las dimensiones de la ventana, la posición de la cámara y las matrices de transformación.

```
// ==============================
// CONSTANTS & GLOBAL VARIABLES
// ==============================
const GLuint WIDTH = 1920, HEIGHT = 1080;
int SCREEN_WIDTH, SCREEN_HEIGHT;

Camera camera(glm::vec3(0.0f, 0.0f, 3.0f));
GLfloat lastX = WIDTH / 2.0f;
GLfloat lastY = HEIGHT / 2.0f;
bool keys[1024];
bool firstMouse = true;
bool active = false;

GLfloat deltaTime = 0.0f;
GLfloat lastFrame = 0.0f;

glm::vec3 lightPos(0.0f);
glm::vec3 Light1(0.0f);
glm::vec3 pointLightPositions[4] = {
    glm::vec3(0.0f), glm::vec3(0.0f),
    glm::vec3(0.0f), glm::vec3(0.0f)
};
```

Después, empieza la declaración de la función principal del programa, creando la ventana donde se mostrará la escena.

```cpp
int main()
{
    // Init GLFW
    glfwInit();

    // Create a GLFWwindow object that we can use for GLFW's functions
    GLFWwindow* window = glfwCreateWindow(WIDTH, HEIGHT, "Fuentes de luz", nullptr, nullptr);

    if (nullptr == window)
    {
        std::cout << "Failed to create GLFW window" << std::endl;
        glfwTerminate();

        return EXIT_FAILURE;
    }

    glfwMakeContextCurrent(window);

    glfwGetFramebufferSize(window, &SCREEN_WIDTH, &SCREEN_HEIGHT);

    // Set the required callback functions
    glfwSetKeyCallback(window, KeyCallback);
    glfwSetCursorPosCallback(window, MouseCallback);
```

Luego, en el main, cargamos los shaders de iluminación

```cpp
Shader lightingShader("Shader/lighting.vs", "Shader/lighting.frag");
Shader lampShader("Shader/lamp.vs", "Shader/lamp.frag");
```

Posteriormente, cargamos los modelos que se van a observar en la escena

```cpp
//Carga de los modelos
//Sala
Model Sillon((char*)"ProyectoBonito/Sala/Sillon.obj");
Model Tele((char*)"ProyectoBonito/Sala/Tele.obj");
Model AlfombraRoja((char*)"ProyectoBonito/Sala/AlfombraRoja.obj");
Model BaseLampara((char*)"ProyectoBonito/Sala/BaseLampara.obj");
Model Lampara((char*)"ProyectoBonito/Sala/Lampara.obj");
Model Foco((char*)"ProyectoBonito/Sala/Foco.obj");
Model LamparaDet((char*)"ProyectoBonito/Sala/Foco.obj");
Model LampB1((char*)"ProyectoBonito/Sala/LampB1.obj");
Model LampB2((char*)"ProyectoBonito/Sala/LampB2.obj");
Model LamparaHolder((char*)"ProyectoBonito/Sala/LampHolder.obj");
Model MesaCentro((char*)"ProyectoBonito/Sala/MesaCentro.obj");
Model AlfombraBlanca((char*)"ProyectoBonito/Sala/AlfombraBlanca.obj");

//Exterior
Model Chimenea((char*)"ProyectoBonito/Exterior/Chimenea.obj");
Model MarcoVentanaDelantera((char*)"ProyectoBonito/Exterior/MarcoVentanaDelantera.obj");
Model MarcoVentanaLateral((char*)"ProyectoBonito/Exterior/MarcoVentanaLateral.obj");
```

Se declaran y configuran los buffers

```
GLuint VBO, VAO;
glGenVertexArrays(1, &VAO);
glGenBuffers(1, &VBO);
glBindVertexArray(VAO);
glBindBuffer(GL_ARRAY_BUFFER, VBO);
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);
```

Se le da profundidad a la escena declarando la matriz de cámara, calcula la proyección en perspectiva.

```
glm::mat4 projection = glm::perspective(camera.GetZoom(), (GLfloat)SCREEN_WIDTH / (GLfloat)SCREEN_HEIGHT, 0.8f, 50.0f);
```

Se crea el ciclo que corre hasta que se cierra la ventana.

```
// Game loop
while (!glfwWindowShouldClose(window))
{

    // Calculate deltatime of current frame
    GLfloat currentFrame = glfwGetTime();
    deltaTime = currentFrame - lastFrame;
    lastFrame = currentFrame;

    // Check if any events have been activiated (key pressed, mouse moved etc.) and call corresponding response functions
    glfwPollEvents();
    DoMovement();

    // Clear the colorbuffer
    glClearColor(0.53f, 0.81f, 0.92f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // OpenGL options
    glEnable(GL_DEPTH_TEST);


    // Use cooresponding shader when setting uniforms/drawing objects
    lightingShader.Use();
```

Este ciclo se encarga de limpiar el buffer de pantalla, enviar posición de la cámara y se dibujan los modelos tridimensionales para agregarlos a la escena.

```
//Dibujo de modelos

model = glm::mat4(1);
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Sillon.Draw(lightingShader);

model = glm::mat4(1);
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Tele.Draw(lightingShader);

model = glm::mat4(1);
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
AlfombraRoja.Draw(lightingShader);

model = glm::mat4(1);
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
BaseLampara.Draw(lightingShader);

model = glm::mat4(1);
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Lampara.Draw(lightingShader);

model = glm::mat4(1);
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Foco.Draw(lightingShader);
```

Por último, se crean las funciones para el manejo de entradas por parte del usuario.

```
// Camera controls
if (keys[GLFW_KEY_W] || keys[GLFW_KEY_UP])
{
    camera.ProcessKeyboard(FORWARD, deltaTime);

}

if (keys[GLFW_KEY_S] || keys[GLFW_KEY_DOWN])
{
    camera.ProcessKeyboard(BACKWARD, deltaTime);


}

if (keys[GLFW_KEY_A] || keys[GLFW_KEY_LEFT])
{
    camera.ProcessKeyboard(LEFT, deltaTime);


}

if (keys[GLFW_KEY_D] || keys[GLFW_KEY_RIGHT])
{
    camera.ProcessKeyboard(RIGHT, deltaTime);


}
```

Conclusiones

- Se logró una recreación efectiva de un entorno ficticio con alto valor de reconocimiento visual.
- El proceso permitió aplicar técnicas clave de modelado, texturizado y UV mapping en Blender.

- Se desarrolló un flujo de trabajo organizado que facilitó la producción eficiente del entorno completo.
- Este proyecto sirvió como experiencia integradora de habilidades en diseño 3D, planificación técnica y documentación profesional.