



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Laboratorio de Computación gráfica e interacción
humano computadora



Proyecto Final

PROFESOR: Ing. Edén Espinoza Urzúa

ALUMNO: Martínez Pérez Brian Erik

GRUPO: 04

SEMESTRE: 2026-1

ENTREGA: 15 de noviembre de 2025

MANUAL TECNICO



ÍNDICE

Contenido

1.- Objetivos.....	3
2.- Diagrama de Flujo del software.....	4
3.- Diagrama de Gantt.....	8
4.- Alcance del proyecto.....	8
5.- Limitantes	10
6.- Metodología	12
7. – Documentación del Código	13
8. – Conclusiones	22
9. – Referencias.....	22



1.- Objetivos

- El alumno deberá aplicar y demostrar los conocimientos adquiridos durante el curso mediante la creación de una recreación 3D en OpenGL.
- Lograr recrear la fachada de la casa del protagonista del juego “Grand Theft Auto San Andreas”.
- Modelar 7 objetos en Open GL que tengan coherencia con el ambiente 3D generado.
- Implementar 3 animaciones básicas y 2 animaciones complejas utilizando lógica del propio lenguaje. Además de que cada animación debe estar en el contexto del ambiente.
- Realizar entradas de teclado, para que el usuario pueda interactuar con el ambiente 3D generado.

2.- Diagrama de Flujo del software

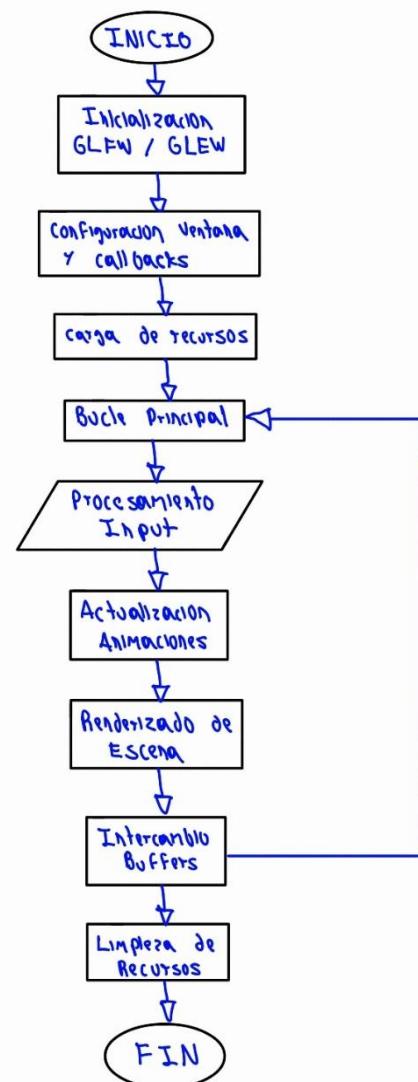


Imagen 1.1 – Diagrama de flujo de la ejecución del proyecto.



Si $\text{rotacionPuerta} == 0^\circ$ entonces
 $\text{rotacionPuerta} = 90^\circ$

De lo contrario

$\text{rotacionPuerta} = 0^\circ$

FIN SI

Imagen 1.2 – pseudocódigo de la animación “Abrir o cerra puerta”.

Si $\text{rotacionPuerta} \leq 0^\circ$ AND $\text{puertaArriba} == \text{falso}$ entonces
Disminuir RotacionPuerta 0.5 unidades
Si $\text{rotacionPuerta} < -20.5^\circ$ entonces
PuertaArriba es Verdadero
FIN SI
De lo contrario
Aumentar RotacionPuerta 0.5 unidades
Si $\text{rotacionPuerta} > -0.5^\circ$ entonces
PuertaArriba es Falso
FIN SI
FIN SI

Imagen 1.3 – pseudocódigo de la animación “Abrir o cerrar puerta del garage”.



```
Si PosicionCortina == 0 AND CortinaAbierta == falso entonces
    Aumentar PosicionCortina 0.5 unidades
    Si PosicionCortina > 0.7 entonces
        CortinaAbierta es Verdadero
    Fin Si
De lo contrario
    0.5 unidades
    Si PosicionCortina <= 0.01 entonces
        CortinaAbierta es Falso
    Fin Si
Fin Si
```

Imagen 1.4 – pseudocódigo de la animación “Abrir o cerrar cortina”.

```
Si Activo == Verdadero entonces
    Activar Luces Blancas
De lo contrario
    Desactivar Luces Blancas
Fin Si
```

Imagen 1.5 – pseudocódigo de la animación “Prender o apagar luces”.

Aumentar Rotacion de carro 0.1 unidades.

Aumentar timer "deltatime" veces.

Posicionllanta es 0.05 unidades.

Si timer >= Intervalo - cambio **entonces**

Posicionllanta es -0.05 unidades.

Decrementar timer "Intervalo - cambio" veces.

Fin Si

Imagen 1.6 – pseudocódigo de la animación “Avanzar carro”.

Si rotacion == 0 **entonces**

Mover personaje en el eje "z"

Rotar articulacion en sentido positivo

De lo contrario

Mover personaje en el eje "z"

Rotar articulacion en sentido negativo

Fin Si

Si personaje.z == 0.5 **entonces**

Rotar personaje 90° en el eje "y"

Fin Si

Si rotacion.z == 90° **entonces**

Mover personaje en el eje "x"

Rotar articulacion en sentido positivo

De lo contrario

Mover personaje en el eje "x"

Rotar articulacion en sentido negativo

Fin Si

Si personaje.x > 4.6 **entonces**

Detener Animacion

Fin Si

Imagen 1.7 – pseudocódigo de la animación “movimiento personaje”.

3.- Diagrama de Gantt

	22-sep	23-SEP 29-SEP	30-SEP 7-oct	8-oct 15-oct	16-oct 23-oct	24-oct 31-oct	1-nov 5-nov	6-nov 12-nov	13-nov 14-nov	15-nov
Inicio de proyecto	1 Dia									
Decision de fachada a implementar		7 Dias								
Visualizacion de contenido realacionado a "Blender"			8 Dias							
Implementacion de la fachada en Blender				8 Dias						
Texturizado de la fachada en blender					8 Dias					
Integracion del modelo generado en Blender en OpenGL						8 Dias				
Creacion y cargado de modelos en OpenGL							5 Dias			
Implementacion de animaciones sencillas y complejas								7 Dias		
Documentacion del proyecto									2 Dias	
Fin del proyecto										1 Dia

Imagen 2.1 – Diagrama de Gantt de las actividades realizadas durante el proyecto.

4.- Alcance del proyecto

En este proyecto solo se queda hasta el diseño por fuera de la casa del protagonista del juego “Grand Theft Auto San Andreas”, en este caso me enfoque en desarrollar la vista frontal de la casa. Que seria la parte de la puerta blanca y el pequeño pasillo que se encuentra en el lado izquierdo. Además de la sección del garaje que pertenece a la casa del protagonista del juego.



Imagen 2.2 – referencia de la fachada creada.



Imagen 2.3 – Modelo de la fachada implementada en el proyecto.

Ya para el interior de la casa solo desarrolle 2 cuartos, la cocina y la sala. La cocina contiene los elementos físicos visuales mas importantes, como los son la estufa, refrigerador y guarda platos que fueron hechos en Open GL. Los demás elementos fueron modelados desde Blender o descargados desde Sketchfab y cargados a Open GL.



Imagen 2.4 – referencia de la cocina de la fachada.



Imagen 2.5 – Modelo de la cocina realizada en el proyecto.

El ultimo cuarto implementado fue la sala de la casa, este fue el cuarto más difícil de amueblar ya que tenía una mayor cantidad de elementos. y muchos de los elementos eran tardados de modelar, por lo que decidí solo crear los objetos con open gl, los más fáciles. En este caso yo modele con open gl la planta ubicada en la entrada de la cocina, una mesa, una silla y el cuadro fotográfico que contiene la foto de un paisaje del propio juego. Las demás fotografías y mesas para la planta las realice en blender, y los demás objetos son modelos descargados de Sketchfab.



Imagen 2.6 – referencia de la sala de la fachada.



Imagen 2.7 – Modelo de la sala realizada en el proyecto.

5.- Limitantes

Las limitaciones del proyecto son notorias ya que decidí omitir los cuartos de arriba de la casa, dentro de la casa solo tengo las escaleras de adorno, ya que no llevan a ningún lado y chocan con las paredes de la casa.

Para este proyecto decidí solo enfocarme en los 2 cuartos del primer piso ya que son los primeros que conocemos en el juego, son los mas populares, que causan una sensación de recordarlos más fácil y además son los más reconocidos de la casa por las personas que jugaron el juego.



Imagen 3.1 – escaleras de adorno implementadas.



Imagen 3.2 – Escaleras del juego que llevan al piso de arriba, NO implementadas en este proyecto.



Imagen 3.3 – Parte de arriba de la casa vacía, solo para adornar el exterior de la fachada.



Imagen 3.4 – Cuartos del piso de arriba que NO fueron implementados en el proyecto.

Otra característica para tomar en cuenta fue la computadora en la que desarrolle el proyecto. Ya que los recursos con los que cuenta mi computadora son decentes. Ya que no cuento con una tarjeta gráfica en mi computadora, por lo que mi procesador y mi memoria RAM se destinan en el procesamiento de gráficos y vuelve más lento el proceso de compilación del proyecto.

Debido a esta limitante en los recursos de mi computadora, me retrase en el cargado de modelos en open gl, además de que a la hora de agregar los modelos descargados. La compilación del proyecto duraba hasta minuto y medio en ejecutarse, por lo que volvió lento el desarrollo del proyecto.

 Almacenamiento 1.39 TB 347 GB de 1.39 TB usado	 Tarjeta gráfica 2 GB AMD Radeon(TM) Graphics	 RAM instalada 16.0 GB Velocidad: 2400 MT/s	 Procesador AMD Ryzen 5 3450U with Radeon Vega Mobile Gfx 2.10 GHz
---	---	---	--

Imagen 3.5 – recursos limitados de la computadora con la que realice el proyecto.

6.- Metodología

La metodología que aplique en el proyecto fue Waterfall (cascada) es una metodología en la que las etapas se organizan de arriba a abajo. Se desarrollan las



diferentes funciones en etapas diferenciadas y obedeciendo un riguroso orden. Antes de cada etapa se debe revisar el producto para ver si está listo para pasar a la siguiente fase. Los requisitos y especificaciones iniciales no están predispuestos para cambiarse, por lo que no se pueden ver los resultados hasta que el proyecto ya esté bastante avanzado.

Es por lo que en el diagrama de Gantt las actividades realizadas en el proyecto son secuenciales, por eso primero decidí que fachada implementar, luego aprender a modelar en blender, y proceder a modelar la casa, texturizarla y cargarla con las librerías de open gl. Para dejar para después el proceso de amueblar la casa con modelos descargados. Por último, en el Código, implementar las animaciones solicitadas, ya teniendo en cuenta los objetos con los que cuento y el espacio 3D donde se realizaran cada una.

7. – Documentación del Código

Includes y Configuración Inicial

```
#include <iostream>
#include <cmath>
// GLEW
#include <GL/glew.h>
// GLFW
#include <GLFW/glfw3.h>
// Other Libs
#include "stb_image.h"
// GLM Mathematics
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>
//Load Models
#include "SOIL2/SOIL2.h"
// Other includes
#include "Shader.h"
#include "Camera.h"
#include "Model.h"
```

Imagen 4.1 - Includes y Configuración Inicial

```
// Window dimensions
const GLuint WIDTH = 1920, HEIGHT = 1080;
int SCREEN_WIDTH, SCREEN_HEIGHT;
```

Imagen 4.2 - Configuración de Ventana



Configuración e implementación de la cámara sintética dentro del proyecto realizado.

```
// Camera
Camera camera(glm::vec3(-2.0f, 0.0f, 18.0f)); //posición inicial de la cámara
GLfloat lastX = WIDTH / 2.0;
GLfloat lastY = HEIGHT / 2.0;
bool keys[1024];
bool firstMouse = true;
```

Imagen 4.3 – sistema de cámara sintética.

```
// Positions of the point lights
glm::vec3 pointLightPositions[] = {
    glm::vec3(1.6f, -1.7f, 0.67f),      // Luz 1
    glm::vec3(-0.4f, 4.46f, 0.64f),     // Luz 2
};
```

Imagen 4.4 – declaración de la ubicación de las luces puntuales.

```
//Variables de Animacion
float rotCarro = 0.0f;
float rotPuerta = 0.0f;
float rotPuertaGarage = 0.0f;
float posCortina = 0.0f;
float tamllanta = 0.0f;
float timer = 10.0f; // Acumulador de tiempo
float FLegs = 0.0f;
float RLegs = 0.0f;
float personajePosX = 0.0f;
float personajePosZ = -2.0f;
float personajeRot = 0.0f;
float posPuertaX = 0.0f;
float posPuertaZ = 0.0f;
const float INTERVALO_CAMBIO = 0.1f; // Segundos que deben pasar antes del cambio

// Variables para Activar o Desactivas animaciones
bool AnimCarro = false;
bool AnimPuerta = false;
bool AnimGarage = false;
bool AnimCortina = false;
bool AnimPersonaje = false;
bool puertaArriba = false;
bool cortinaAbierta = false;
bool step = false;
```

Imagen 4.5 – variables de animación utilizadas.



```
//dibujar
glm::mat4 Dibujar(glm::mat4 model, glm::vec3 escala, glm::vec3 traslado, GLint uniformModel) {
    glm::mat4 modelTemp = model; // copia la transformación actual
    modelTemp = glm::translate(modelTemp, traslado);
    model = glm::scale(model, escala);
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    glDrawArrays(GL_TRIANGLES, 0, 36);

    return modelTemp; // devolvemos la transformación
}
//dibujar con rotacion
glm::mat4 DibujarR(glm::mat4 model, glm::vec3 escala, glm::vec3 traslado, glm::vec3 rotacion, float angulo, GLint uniformModel) {
    glm::mat4 modelTemp = model; // copia la transformación actual
    modelTemp = glm::translate(modelTemp, traslado);
    model = glm::rotate(model, glm::radians(angulo), rotacion);
    model = glm::scale(model, escala);

    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    glDrawArrays(GL_TRIANGLES, 0, 36);

    return modelTemp; // devolvemos la transformación
}
//dibujar con rotacion
glm::mat4 DibujarC(glm::mat4 model, glm::vec3 escala, glm::vec3 traslado, glm::vec3 rotacion, float angulo, GLint uniformModel, int cara) {
    glm::mat4 modelTemp = model; // copia la transformación actual
    modelTemp = glm::translate(modelTemp, traslado);
    model = glm::rotate(model, glm::radians(angulo), rotacion);
    model = glm::scale(model, escala);

    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    glDrawArrays(GL_TRIANGLES, 0, cara);

    return modelTemp; // devolvemos la transformación
}
```

Imagen 4.6 - Funciones de dibujo, utilizadas en los modelos creados en Open Gl.

Carga de Texturas, habilita blending (canal alfa), carga imágenes usando stb_image, configura parámetros de textura (wrap, filtering), genera mipmaps automáticamente.

```
// FUNCIÓN PARA CARGA DE TEXTURAS
GLuint LoadTexture(const char* path) {
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    GLuint textureID;
    glGenTextures(1, &textureID);
    glBindTexture(GL_TEXTURE_2D, textureID);
    int textureWidth, textureHeight, nrChannels;
    stbi_set_flip_vertically_on_load(true);
    unsigned char* image;
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST_MIPMAP_NEAREST);
    // Diffuse map
    image = stbi_load(path, &textureWidth, &textureHeight, &nrChannels, 0);
    glBindTexture(GL_TEXTURE_2D, textureID);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, textureWidth, textureHeight, 0, GL_RGB, GL_UNSIGNED_BYTE, image);
    glGenerateMipmap(GL_TEXTURE_2D);
    if (image) {
        glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, textureWidth, textureHeight, 0, GL_RGB, GL_UNSIGNED_BYTE, image);
        glGenerateMipmap(GL_TEXTURE_2D);
    }
    else {
        std::cout << "Error al cargar textura: " << path << std::endl;
    }
    stbi_image_free(image);
    return textureID;
}
```

Imagen 4.7 – función para cargar texturas de los modelos.

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

Imagen 4.8 – shaders utilizados en el proyecto.

Carga de modelos utilizados en el proyecto.

```
//modelos utilizados en el exterior de la casa  
Model Piso_Exterior((char*)"Models/piso_exterior.obj");  
Model Skybox((char*)"Models/skybox.obj");
```

Imagen 4.9 – carga de modelos utilizados para la ambientación.

```
//modelos de la casa  
Model Casa((char*)"Models/casa.obj");  
Model Tranparente((char*)"Models/objetos_trans.obj");  
Model Puerta((char*)"Models/puerta.obj");  
Model PuertaGarage((char*)"Models/puerta_garage.obj");  
Model Cortina_Der((char*)"Models/cortina_der.obj");  
Model Cortina_Izq((char*)"Models/cortina_izq.obj");
```

Imagen 4.10 – carga de modelos utilizados para la fachada.

```
//Cocina  
Model Complemento_Cocina((char*)"Modelos_Cocina/complemento_cocina.obj");  
Model Maceta((char*)"Modelos_Cocina/maceta.obj");  
Model Alcohol((char*)"Modelos_Cocina/alcohol.obj");  
Model Lata((char*)"Modelos_Cocina/latas.obj");  
Model Plato((char*)"Modelos_Cocina/platos.obj");
```

Imagen 4.11 – carga de modelos utilizados para amueblar la cocina.

```
//Sala  
Model Complemento_Sala((char*)"Modelos_Sala/complemento_sala.obj");  
Model Cajonera((char*)"Modelos_Sala/cajonera.obj");  
Model Revista((char*)"Modelos_Sala/revista.obj");  
Model Florero((char*)"Modelos_Sala/florero.obj");  
Model Television((char*)"Modelos_Sala/television.obj");  
Model Sala((char*)"Modelos_Sala/sala.obj");  
Model Sillon((char*)"Modelos_Sala/sillon.obj");  
Model Silla((char*)"Modelos_Sala/silla.obj");  
Model Taza((char*)"Modelos_Sala/taza.obj");
```

Imagen 4.12 – carga de modelos utilizados para amueblar la sala.

```
// Modelos Animacion 1  
Model Carro((char*)"carro/carro.obj");  
Model Llanta1((char*)"carro/llanta_1.obj");  
Model Llanta2((char*)"carro/llanta_2.obj");  
Model Llanta3((char*)"carro/llanta_3.obj");  
Model Llanta4((char*)"carro/llanta_4.obj");  
  
//modelos Aniamcion 2  
Model Torso((char*)"personaje/torso.obj");  
Model Mano_Der((char*)"personaje/mano_der.obj");  
Model Mano_Izq((char*)"personaje/mano_izq.obj");  
Model Pie_Der((char*)"personaje/pie_der.obj");  
Model Pie_Izq((char*)"personaje/pie_izq.obj");
```

Imagen 4.13 – carga de los modelos para las animaciones complejas.

```
//Carga de textura
GLuint madera = loadTexture("images/madera.jpg");
GLuint marco = loadTexture("images/marco.jpg");
GLuint hoja = loadTexture("images/hoja.jpg");
GLuint rojo = loadTexture("images/rojo.png");
GLuint negro = loadTexture("images/negro.jpg");
GLuint marron = loadTexture("images/marron.jpg");
GLuint amarillo = loadTexture("images/amarillo.jpg");
GLuint tronco = loadTexture("images/tronco.jpg");
GLuint gris = loadTexture("images/gris.jpg");
GLuint foto = loadTexture("images/foto.jpg");
GLuint mostasa = loadTexture("images/mostasa.jpg");
```

Imagen 4.14 – cargado de texturas para los modelos hechos con Open GL.

```
// -----
// MODELADO CON OPENGL
// -----
```

```
// Mesa
view = camera.GetViewMatrix();
glUniformMatrix4fv(viewLoc, 1, GL_FALSE, glm::value_ptr(view));
glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr(projection));
model = glm::mat4(1.0f);
modelTemp = glm::mat4(1.0f);
modelTemp1 = glm::mat4(1.0f);
// Tabla
modelTemp1 = modelTemp = model = glm::translate(model, glm::vec3(2.5f, -0.25f, -1.2f));
model = glm::scale(model, glm::vec3(2.0f, 0.1f, 1.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glDrawArrays(GL_TRIANGLES, 0, 36);
//Patas Tabla
modelTemp = Dibujar(modelTemp, glm::vec3(0.1f, 0.9f, 0.1f), glm::vec3(0.8f, -0.5f, 0.4f), modelLoc);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.1f, 0.9f, 0.1f), glm::vec3(0.8f, -0.5f, -0.4f), modelLoc);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.1f, 0.9f, 0.1f), glm::vec3(-0.8f, -0.5f, 0.4f), modelLoc);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.1f, 0.9f, 0.1f), glm::vec3(-0.8f, -0.5f, -0.4f), modelLoc);

//Silla
modelTemp = Dibujar(modelTemp1, glm::vec3(0.1f, 0.5f, 0.1f), glm::vec3(1.5f, -0.7f, 0.35f), modelLoc); //patas
modelTemp = Dibujar(modelTemp1, glm::vec3(0.1f, 0.5f, 0.1f), glm::vec3(1.5f, -0.7f, -0.35f), modelLoc);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.1f, 0.5f, 0.1f), glm::vec3(2.1f, -0.7f, 0.35f), modelLoc);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.1f, 0.5f, 0.1f), glm::vec3(2.1f, -0.7f, -0.35f), modelLoc);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.7f, 0.1f, 0.8f), glm::vec3(1.8f, -0.4f, 0.0f), modelLoc); //asiento
modelTemp = Dibujar(modelTemp1, glm::vec3(0.1f, 0.7f, 0.8f), glm::vec3(2.1f, 0.0f, 0.0f), modelLoc); //respaldo
```

```
// Refrigerador:
model = glm::mat4(1.0f);
glBindTexture(GL_TEXTURE_2D, amarillo); // textura blanca
modelTemp1 = modelTemp = Dibujar(model, glm::vec3(0.69f, 1.5f, 0.6f), glm::vec3(-1.31f, -0.5f, -6.3f), modelLoc); // Puerta superior
glBindTexture(GL_TEXTURE_2D, gris);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.5f, 0.6f, 0.005f), glm::vec3(0.0f, 0.4f, 0.31f), modelLoc); // Puerta superior
modelTemp = Dibujar(modelTemp1, glm::vec3(0.5f, 0.3f, 0.005f), glm::vec3(0.0f, -0.2f, 0.31f), modelLoc); // Puerta inferior
glBindTexture(GL_TEXTURE_2D, negro);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.04f, 0.3f, 0.05f), glm::vec3(0.20f, 0.4f, 0.35f), modelLoc); // Mango puerta superior
modelTemp = Dibujar(modelTemp1, glm::vec3(0.04f, 0.15f, 0.05f), glm::vec3(0.20f, -0.2f, 0.35f), modelLoc); // Mango puerta inferior

//estufa
model = glm::mat4(1.0f);
glBindTexture(GL_TEXTURE_2D, negro);
modelTemp1 = modelTemp = Dibujar(model, glm::vec3(0.7f, 0.8f, 0.6f), glm::vec3(-1.9f, -0.6f, -2.76f), modelLoc); // soporte superior
glBindTexture(GL_TEXTURE_2D, gris);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.7f, 0.2f, 0.1f), glm::vec3(0.0f, 0.5f, 0.25f), modelLoc); // soporte superior
modelTemp = Dibujar(modelTemp1, glm::vec3(0.5f, 0.5f, 0.005f), glm::vec3(0.0f, -0.1f, -0.31f), modelLoc); // Puerta estufa
modelTemp = Dibujar(modelTemp1, glm::vec3(0.15f, 0.07f, 0.05f), glm::vec3(0.2f, 0.3f, -0.31f), modelLoc); // manija estufa
modelTemp = Dibujar(modelTemp1, glm::vec3(0.15f, 0.07f, 0.05f), glm::vec3(-0.2f, 0.3f, -0.31f), modelLoc); // manija estufa
glBindTexture(GL_TEXTURE_2D, rojo);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.13f, 0.05f, 0.13f), glm::vec3(0.2f, 0.4f, 0.08f), modelLoc); // parrilla estufa
modelTemp = Dibujar(modelTemp1, glm::vec3(0.13f, 0.05f, 0.13f), glm::vec3(-0.2f, 0.4f, 0.08f), modelLoc); // parrilla estufa
modelTemp = Dibujar(modelTemp1, glm::vec3(0.13f, 0.05f, 0.13f), glm::vec3(0.2f, 0.4f, -0.15f), modelLoc); // parrilla estufa
modelTemp = Dibujar(modelTemp1, glm::vec3(0.13f, 0.05f, 0.13f), glm::vec3(-0.2f, 0.4f, -0.15f), modelLoc); // parrilla estufa
```

```

//planta en maceta
model = glm::mat4(1.0f);
glBindTexture(GL_TEXTURE_2D, marron);
modelTemp1 = modelTemp = Dibujar(model, glm::vec3(0.3f, 0.4f, 0.3f), glm::vec3(-1.0f, -0.8f, -1.9f), modelLoc);
glBindTexture(GL_TEXTURE_2D, tronco);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.09f, 1.5f, 0.09f), glm::vec3(0.0f, 0.2f, 0.0f), modelLoc); //tallo
glBindTexture(GL_TEXTURE_2D, hoja);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.25f, 0.3f, 0.25f), glm::vec3(0.0f, 0.8f, 0.0f), modelLoc); //hoja superior
modelTemp = Dibujar(modelTemp1, glm::vec3(0.25f, 0.15f, 0.05f), glm::vec3(0.0f, 0.5f, 0.0f), modelLoc); //hoja media
modelTemp = Dibujar(modelTemp1, glm::vec3(0.25f, 0.15f, 0.05f), glm::vec3(0.0f, 0.3f, 0.0f), modelLoc); //hoja inferior

//Cuadro fotográfico
model = glm::mat4(1.0f);
glBindTexture(GL_TEXTURE_2D, marco);
modelTemp1 = modelTemp = Dibujar(model, glm::vec3(0.05f, 0.4f, 0.55f), glm::vec3(4.8f, 0.4f, 3.5f), modelLoc);
glBindTexture(GL_TEXTURE_2D, foto);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.02f, 0.35f, 0.50f), glm::vec3(-0.05f, 0.0f, 0.0f), modelLoc); //Foto

// guardaplatos
model = glm::mat4(1.0f);
glBindTexture(GL_TEXTURE_2D, mostaza);
modelTemp1 = modelTemp = Dibujar(model, glm::vec3(0.25f, 0.8f, 0.7f), glm::vec3(-3.4f, 0.4f, -3.2f), modelLoc);
glBindTexture(GL_TEXTURE_2D, marron);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.02f, 0.7f, 0.25f), glm::vec3(0.18f, 0.0f, 0.15f), modelLoc); //puerco
modelTemp = Dibujar(modelTemp1, glm::vec3(0.02f, 0.7f, 0.25f), glm::vec3(0.18f, 0.0f, -0.15f), modelLoc); //puerco
glBindTexture(GL_TEXTURE_2D, negro);
modelTemp = Dibujar(modelTemp1, glm::vec3(0.05f, 0.15f, 0.05f), glm::vec3(0.18f, 0.0f, 0.1f), modelLoc); // Mangrullo
modelTemp = Dibujar(modelTemp1, glm::vec3(0.05f, 0.15f, 0.05f), glm::vec3(0.18f, 0.0f, -0.1f), modelLoc); // Mangrullo

```

Imagen 4.15 – modelado de 7 objetos con texturas en Open GL.

```

// Moves/alters the camera positions based on user input
void DoMovement()
{
    // 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);
    }
}

```

Imagen 4.16 – teclas para la manipulación de la cámara sintética.

```

if (keys[GLFW_KEY_SPACE])
{
    active = !active;
    if (active)
    {
        Light1 = glm::vec3(1.0f, 1.0f, 1.0f);
    }
    else
    {
        Light1 = glm::vec3(0); //Cuando es solo un valor en los 3 vectores pueden dejar solo una componente
    }
}

```

Imagen 4.17 – tecla para activar o desactivar sistema de iluminación interno.

```

//Animacion Abrir la puerta
if (keys[GLFW_KEY_Z])
{
    AnimPuerta = !AnimPuerta;
}

//Animacion Abrir puerta de Garage
if (keys[GLFW_KEY_X])
{
    AnimGarage = !AnimGarage;
}

//Animacion para abrir o cerrar cortinas
if (keys[GLFW_KEY_C])
{
    AnimCortina = !AnimCortina;
}

//Animacion para mover el automovil
if (keys[GLFW_KEY_V])
{
    AnimCarro = !AnimCarro;
}

//Animacion para mover el personaje
if (keys[GLFW_KEY_B])
{
    AnimPersonaje = !AnimPersonaje;
}

```

Imagen 4.18 – teclas para activar o desactivar las animaciones implementadas.

```

// animacion para hacer caminar el personaje dentro de la casa
if (AnimPersonaje)
{
    if (personajeRot == 0.0f)
    {
        if (!step) //state 1
        {
            RLegs += 0.03f;
            FLegs += 0.03f;

            if (RLegs > 5.0f) { //condition
                step = true;
            }
            personajePosZ += 0.001;
        }
        else
        {
            RLegs -= 0.03f;
            FLegs -= 0.03f;

            if (RLegs < -5.0f) { //condition
                step = false;
            }
            personajePosZ += 0.001;
        }
    }

    if (personajePosZ > 0.5f)
    {
        personajeRot = 90.0f;
        //AnimPuerta = true;
    }
}

if (personajeRot == 90.0f)
{
    if (!step) //state 1
    {
        RLegs += 0.03f;
        FLegs += 0.03f;

        if (RLegs > 5.0f) { //condition
            step = true;
        }
        personajePosX += 0.001;
    }
    else
    {
        RLegs -= 0.03f;
        FLegs -= 0.03f;

        if (RLegs < -5.0f) { //condition
            step = false;
        }
        personajePosX += 0.001;
    }
}

if (personajePosX > 4.60584f)
{
    AnimPersonaje = !AnimPersonaje;
}

```

Imagen 4.19 – lógica de la animación “movimiento personaje”.

```
//Animacion Abrir la puerta de la casa
if (AnimPuerta)
{
    if (rotPuerta == 0.0f)
    {
        rotPuerta = 90.0f;
        posPuertaX = -6.0f;
        posPuertaZ = -6.25f;
        AnimPuerta = !AnimPuerta;
    }
    else
    {
        rotPuerta = 0.0f;
        posPuertaX = 0.0f;
        posPuertaZ = 0.0f;
        AnimPuerta = !AnimPuerta;
    }
}
```

Imagen 4.20 – lógica de la animación “Abrir o cerra puerta”.

```
//Animacion para abrir la puerta del garage
if (AnimGarage)
{
    if (rotPuertaGarage <= 0.0f and puertaArriba==false)
    {
        rotPuertaGarage -= 0.5;
        if (rotPuertaGarage < -20.5f)
        {
            AnimGarage = !AnimGarage;
            puertaArriba = true;
        }
    }
    else
    {
        rotPuertaGarage += 0.5;
        if (rotPuertaGarage > -0.5f)
        {
            AnimGarage = !AnimGarage;
            puertaArriba = false;
        }
    }
}
```

Imagen 4.21 – lógica de la animación “Abrir o cerrar puerta del garaje”.

```
//Animacion para abrir o cerrar las cortinas
if (AnimCortina)
{
    if (posCortina >= 0.0f and cortinaAbierta == false)
    {
        posCortina += 0.01;
        if (posCortina > 0.7f)
        {
            AnimCortina = !AnimCortina;
            cortinaAbierta = true;
        }
    }
    else
    {
        posCortina -= 0.01;
        if (posCortina < 0.1f)
        {
            AnimCortina = !AnimCortina;
            cortinaAbierta = false;
        }
    }
}
```

Imagen 4.22 – lógica de la animación “Abrir o cerrar cortina”.

```
// Animacion para mover el automovil
if (AnimCarro)
{
    rotCarro += 0.1f;
    // 1. Acumular el tiempo transcurrido desde el último fotograma
    timer += deltaTime;

    tamllanta = 0.05f;
    if (timer >= INTERVALO_CAMBIO)
    {
        tamllanta = -0.05f;
        // Reiniciar el temporizador restando el intervalo (para manejar
        timer -= INTERVALO_CAMBIO;
    }
}
```

Imagen 4.23 – lógica de la animación “Avanzar carro”.



8. – Conclusiones

En este proyecto final logré integrar cada uno de los conceptos que vimos en cada práctica, para lograr integrar cada una de las funcionalidades, comencé a partir del Código de máquina de estados, ya que lo considere el más completo para utilizarlo y agregar o eliminar funcionalidades. En la realización del proyecto no solamente utilice open gl, tuve que recurrir a otras herramientas de modelado 3D como “blender”, y también de software de manipulación de imágenes como “gimp”, para la creación de las imágenes que utilice como texturas tuve que utilizar la inteligencia artificial, en específico el generador de imágenes de “Gemini”, yo lo proporcione imágenes tomadas por mi celular del juego y le pedía a Gemini que genere las texturas. Y por último para no tener que modelar yo todo el ambiente, fui a descargar modelos de la página “Skechtfab”, los cuales muchos no estaban bien texturizados y yo tuve que manipular sus coordenadas UV para que se pudieran visualizar correctamente en Open Gl. Por último, agradezco que los profesores de teoría y de laboratorio dejaron la implementación libre del proyecto, ya que este proyecto no solamente implica cuestiones técnicas de las herramientas utilizadas, sino que también depende de la creatividad que el mismo alumno coloca en su proyecto.

9. – Referencias

Blender Flow [@BlenderFlow]. (s/f). *Como aplicar texturas fácil !! | | Blender 3.1 [[Object Object]]*. Youtube. Recuperado el 15 de noviembre de 2025, de <https://www.youtube.com/watch?v=ik9Ldysa6zE>

Casa de los Johnson. (s/f). Grand Theft Encyclopedia; Fandom, Inc. Recuperado el 15 de noviembre de 2025, de https://gta.fandom.com/es/wiki/Casa_de_los_Johnson

DansterDev [@DansterDev]. (s/f). *¡Modelar edificios con interior fácil y sencillo en blender! Explicado paso a paso! [[Object Object]]*. Youtube. Recuperado el 15 de noviembre de 2025, de <https://www.youtube.com/watch?v=Q1lillIriAs>

de Costas, A. [@AlejandrodeCostas]. (s/f-a). *Cómo exportar modelo a OBJ con texturas en Blender - Tutorial guardado en formato Wavefront OBJ 3D [[Object Object]]*. Youtube. Recuperado el 15 de noviembre de 2025, de <https://www.youtube.com/watch?v=Oes-ecXv8nc>

de Costas, A. [@AlejandrodeCostas]. (s/f-b). *TUTORIALES BLENDER - Cómo hacer un mapeado de texturas - Mapeo UV Unwrapping en español Parte 12 [[Object Object]]*. Youtube. Recuperado el 15 de noviembre de 2025, de <https://www.youtube.com/watch?v=Ek-Y62D9W4k>



Interactivas y Computación Gráfica, T. [@ArturoVMS]. (s/f). *Integración de GitHub Desktop con el Proyecto de Computación Gráfica* [[Object Object]]. Youtube. Recuperado el 15 de noviembre de 2025, de <https://www.youtube.com/watch?v=XZYgHmnB1vU>

Metodologías de desarrollo de software: ¿qué son? (s/f). Santander Open Academy. Recuperado el 15 de noviembre de 2025, de <https://www.santanderopenacademy.com/es/blog/metodologias-desarrollo-software.html>

Nisakai [@nisakai_420]. (s/f). *GUÍA RÁPIDA: ¿Cómo modelar en Blender 3D? 🤓 | Tutorial de blender en español 🎉* [[Object Object]]. Youtube. Recuperado el 15 de noviembre de 2025, de https://www.youtube.com/watch?v=f_6QN_G3bk

Modelos Descargados

Grand Theft Auto San Andreas - Balla - Download Free 3D model by Vesauq- (@Trevoncuz2.0). (2025, junio 15). : <https://sketchfab.com/3d-models/grand-theft-auto-san-andreas-balla-cd393ab3131c4160a31be25a5c771ae8>

Books and magazines - Download Free 3D model by Naira (@naira001). (2023, diciembre 27).: <https://sketchfab.com/3d-models/books-and-magazines-d0b76eada5bd495abcdfb2b20e6f7ee6>

Ddiaz Design (Director). (2025, enero 4). *1988 Lamborghini Countach 5000 QV - Download Free 3D model by Ddiaz Design (@ddiaz-design)*. : <https://sketchfab.com/3d-models/1988-lamborghini-countach-5000-qv-884741220ecc44a886f9232f7888a7f1>

Deharo, Y. (Director). (2018, julio 26). *20 Liquor bottles - Download Free 3D model by Yannick Deharo (@YannickDeharo)*. : <https://sketchfab.com/3d-models/20-liquor-bottles-4729b1cfa5b74db68924190242f8ac76>

Dish Rack with Dishes - Download Free 3D model by HippoStance. (2020, mayo 6). : <https://sketchfab.com/3d-models/dish-rack-with-dishes-ce69014964624fbb808a028e5c55d88b>

Elena, F. F. (Director). (2020, septiembre 3). *Saintpaulia flowers low-poly - Download Free 3D model by Elena FF (@elenafefor)*. : <https://sketchfab.com/3d-models/saintpaulia-flowers-low-poly-6f78eed71b74755b3fc14f1ac5924ca>



Mesa minimalista para TV - Download Free 3D model by Draxter_pendragon. (2025, julio 16). : <https://sketchfab.com/3d-models/mesa-minimalista-para-tv-badfe2c2c5a7494cac5bd052145f8f94>

Old Television from 90's - Download Free 3D model by Zgon (@Z-gon). (2022, febrero 22). : <https://sketchfab.com/3d-models/old-television-from-90s-5642074408e94782a7f2ee545e846b45>

Plate and cup - Download Free 3D model by morrrtu1o. (2023, noviembre 19). Serrano, V. H. F. (Director). (2020, diciembre 17). : <https://sketchfab.com/3d-models/plate-and-cup-7c2c9df4bcfb4dfbaa98836af4333f9e>

Latas_Coursera - Download Free 3D model by Victor Hugo Franco Serrano (@VictorHugoFrancoSerrano). : <https://sketchfab.com/3d-models/latas-coursera-3768161dbb7b4f91907934c7fcc7961>

Shekh, A. (Director). (2020, octubre 17). *Pine tree - Download Free 3D model by Andriy Shekh (@sheh5262).* : <https://sketchfab.com/3d-models/pine-tree-e52769d653cd4e52a4acff3041961e65>

Sillas - Download Free 3D model by Elbolillo (@Elbolilloduro). (2021, abril 9). : <https://sketchfab.com/3d-models/sillas-c9e85c037309413299b30dd666ab27c7>

Sillón Le Mans - Download Free 3D model by Pablo.Portela. (2020, agosto 13). : <https://sketchfab.com/3d-models/sillon-le-mans-79e6c35685a24da6899d884fa470df54>

Three_Seater_Sofa - Download Free 3D model by Nelesh_surve. (2025, abril 12). : <https://sketchfab.com/3d-models/three-seater-sofa-bedd882faf354f87a28abcf93c718a3d>