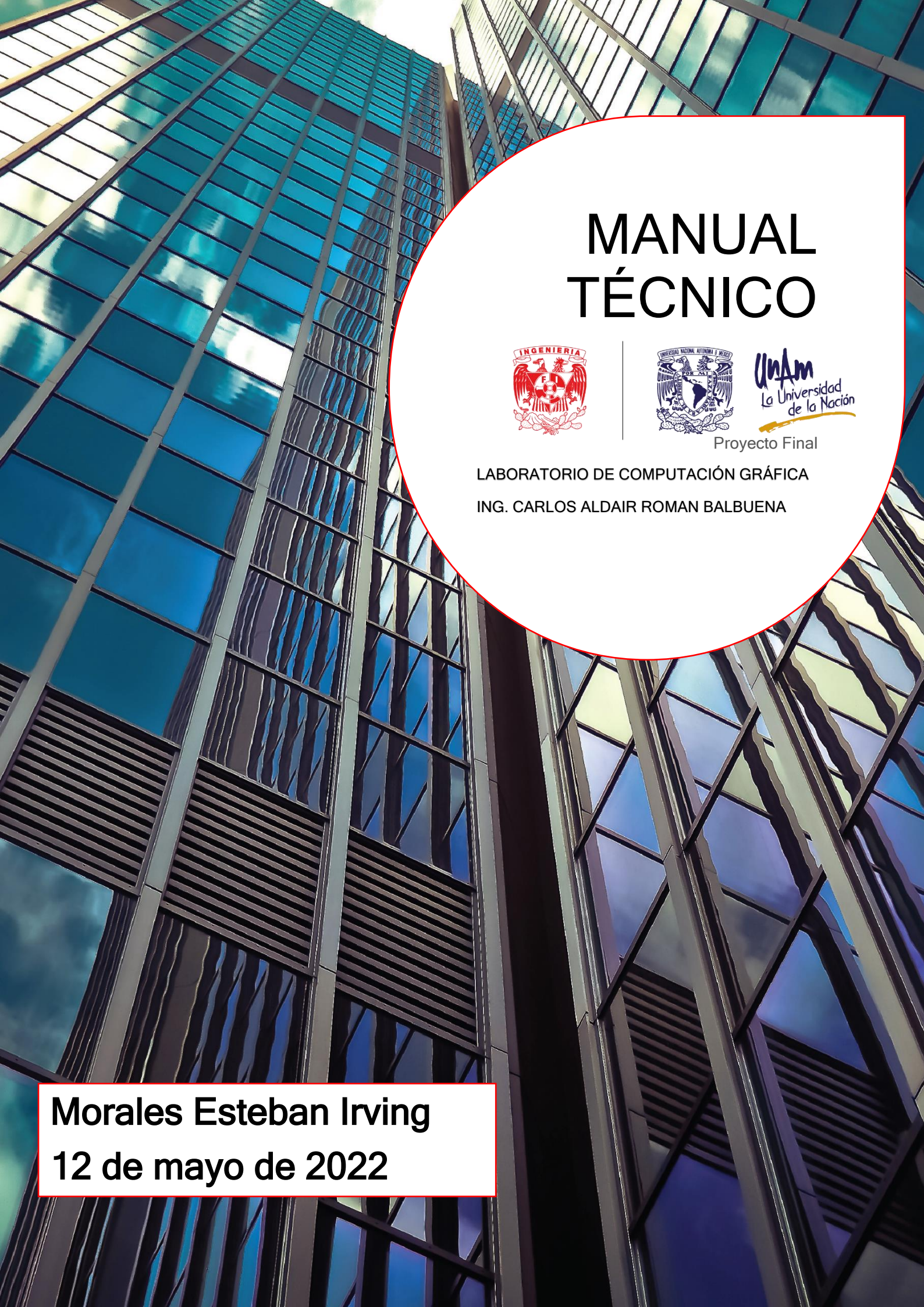


MANUAL TÉCNICO

UNAM La Universidad de la Nación

Proyecto Final

LABORATORIO DE COMPUTACIÓN GRÁFICA

ING. CARLOS ALDAIR ROMAN BALBUENA

Morales Esteban Irving

12 de mayo de 2022

Contenido

Objetivo

El alumno deberá aplicar y demostrar los conocimientos adquiridos durante todo el curso.

El alumno deberá comprender y entender los principios básicos: transformaciones básicas, texturizado y animación.

Diagrama de Gantt

Tareas

1. Aprender a usar Git Hub
2. Aprender a usar GIMP
3. Aprender a usar MAYA
4. Aprender a modelar en Open GL
5. Comenzar con el manual técnico
6. Comenzar con el manual de usuario
7. Retroalimentación sobre el manejo de páginas que pueda usar para descargar modelos gratuitos
8. Comenzar a cargar el repositorio de pruebas
9. Cargar el repositorio del proyecto
10. Modelar objetos de la habitación incluyendo texturas y colores
11. Modelar plantas incluyendo texturizado
12. Modelar paredes incluyendo texturizado
13. Revisar y corregir errores de modelado
14. Ubicar cada objeto sobre el mapa
15. Ubicar las paredes sobre el mapa
16. Revisar errores de ubicación
17. Controlar las animaciones
18. Revisar que no existan problemas de animación
19. Determinar si se agregan elementos extra y ubicarlos sobre el mapa
20. Agregar dichos elementos si los hay
21. Recopilar información sobre lo que se aprendió
22. Retroalimentación de errores
23. Entrega preliminar del proyecto
24. Entrega de proyecto final

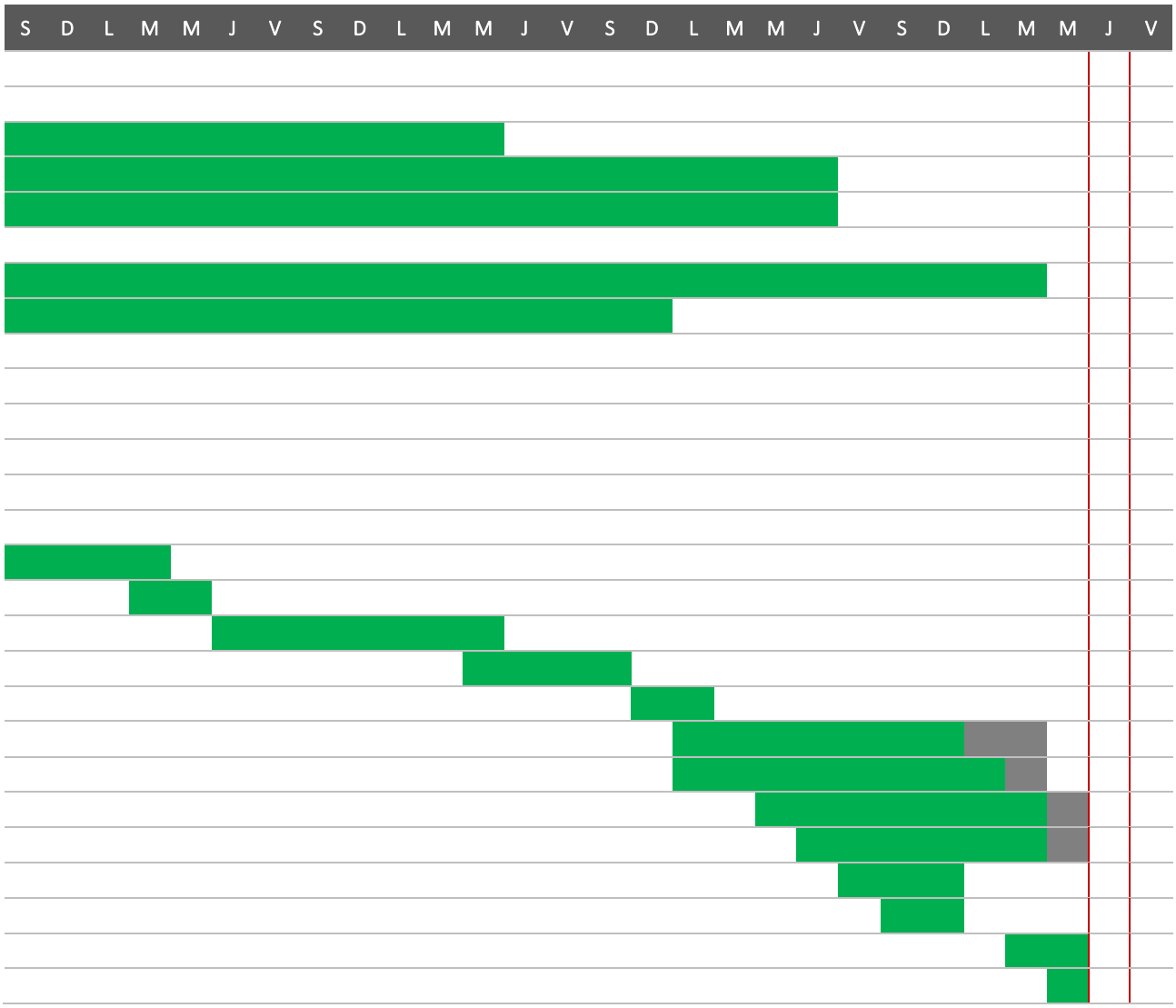
Proyecto FINAL

LIDER DEL PROYECTO: Morales Esteban Irving

				45	26/04/2022
			INICIO DEL PROYECTO		sáb, 12/03/2022

TAREA	RESPONSABLE	PROGRESO	INICIO	FIN
FASE 1				
Tarea 1	M. Irving	100%	20/03/2022	03/04/2022
Tarea 2	M. Irving	100%	20/03/2022	27/04/2022
Tarea 3	M. Irving	100%	12/03/2022	05/05/2022
Tarea4	M. Irving	100%	12/03/2022	05/05/2022
FASE 2	M. Irving			
Tarea 5	M. Irving	100%	23/03/2022	10/05/2022
Tarea 6	M. Irving	100%	23/03/2022	01/05/2022
Tarea 7	M. Irving	100%	24/03/2022	25/03/2022
FASE 3	M. Irving			
Tarea 8	M. Irving	100%	24/03/2022	25/03/2022
Tarea 9	M. Irving	100%	25/03/2022	26/03/2022
Tarea 10	M. Irving	100%	26/03/2022	29/03/2022
Tarea 11	M. Irving	100%	29/03/2022	10/04/2022
Tarea 12	M. Irving	100%	08/04/2022	19/04/2022
Tarea 13	M. Irving	100%	19/04/2022	20/04/2022
Tarea 14	M. Irving	100%	21/04/2022	27/04/2022
Tarea 15	M. Irving	100%	27/04/2022	30/04/2022
Tarea 16	M. Irving	100%	01/05/2022	02/05/2022
Tarea 17	M. Irving	80%	02/05/2022	10/05/2022
Tarea 18	M. Irving	90%	02/05/2022	10/05/2022
Tarea 19	M. Irving	90%	04/05/2022	11/05/2022
Tarea 20	M. Irving	90%	05/05/2022	11/05/2022
Tarea 21	M. Irving	100%	06/05/2022	08/05/2022
Tarea 22	M. Irving	100%	07/05/2022	08/05/2022
Tarea 23	M. Irving	100%	10/05/2022	11/05/2022
Tarea 24	M. Irving	100%	11/05/2022	11/05/2022

sáb, 16/04/2022							sáb, 23/04/2022							sáb, 30/04/2022							sáb, 07/05/2022						
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	01	02	03	04	05	06	07	08	09	10	11	12	13



Alcance del proyecto

Este proyecto es con fines educativos, por lo que se busca comprender funcionalidades básicas, por ello, surgen pequeños fallos como:

- Poca visualización del sky box dentro de la casa habitación, a través de las ventanas, no se visualiza,
- La transparencia sobre cristales solo se aprecia sobre objetos, es decir, cámara->cristal-> objeto.
- La transparencia afecta a algunos objetos que no deberían ser transparentes, depende también del tipo de material que se use sobre el objeto.
- Solo se puede agregar texturas sobre los objetos de tipo Lambert, Phong (color con imagen png, y especular).
- Algunos objetos no se pueden cargar adecuadamente (textura) si se escalan con relación 1000:2

Limitantes

Las limitantes del proyecto son:

- Equipo de cómputo: recursos apenas sostenibles para la carga y ejecución del proyecto.
- Conexión a internet, carga del proyecto, lenta
- Poca optimización sobre los objetos al no mantener un cierto rango de polígonos sobre los mismos.
- La escala, es limitada en la parte visual de la cámara, no se puede realizar un acercamiento adecuado sobre los objetos.

Documentación de código

```cpp
#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"
#include "Texture.h"
#include "modelAnim.h"
```

La importancia de cada una de las librerías es vital mantener un orden,

Cada una de ellas ayuda a:

- Uso de ventana para el entorno grafico
- Visualización de imágenes
- Uso de matrices para contener al objeto
- Carga de modelos por archivos .obj
- Funciones de la cámara
- Transformaciones sobre los objetos
- Carga de textura desde medios externos
- Animaciones por huesos

```cpp
// Function prototypes
void KeyCallback(GLFWwindow* window, int key, int scancode, int action, int mode);
void MouseCallback(GLFWwindow* window, double xPos, double yPos);
void DoMovement();
void animacion();

// Window dimensions
const GLuint WIDTH = 800, HEIGHT = 600;
int SCREEN_WIDTH, SCREEN_HEIGHT;

// Camera
Camera  camera(glm::vec3(0.0f, 5.0f, 5.0f));
GLfloat lastX = WIDTH / 2.0;
GLfloat lastY = HEIGHT / 2.0;
bool keys[1024];
bool firstMouse = true;
float range = 0.0f;
float rot = 0.0f;
```

Las funciones prototipo permiten el manejo de la parte grafica por medio de una ventana y por la cual se tienen un pequeño campo visual del entorno recreado, es importante asignar sus dimensiones.

La parte de la cámara es controlada por su posición inicial y variables que permiten un mejor control por medio del puntero, para simular rotación, zoom y/o uso de esta por medio de un control preestablecido de manera externa.

```
//spootlight
glm::vec3 position(0.0f, 2.0f, 1.0f);
glm::vec3  direction(0.0f, -1.0f, -1.0f);

// Light attributes
glm::vec3 lightPos(0.0f, 0.0f, 0.0f);
//glm::vec3 PosIni(-95.0f, 1.0f, -45.0f);
glm::vec3 PosIni(0.0f, 0.0f, 0.0f);
glm::vec3 lightDirection(0.0f, -1.0f, -1.0f);

bool active;

// Positions of the point lights
glm::vec3 pointLightPositions[] = {
    glm::vec3(1.736f,4.038f, -10.326f),//lamparas cama
    glm::vec3(1.736f,4.038f, - 7.492f),//lamparas cama
    glm::vec3(0.0f,0.0f,  0.0f),
    glm::vec3(0.0f,0.0f, 0.0f)
};
```

una spootlight para simular una lampara externa de alumbrado

se establecen también los puntos iniciales sobre la iluminación: posición y dirección

```
//float segundo = 60.0f, minuto = 60.0f, hora = 12.0f;
bool banderaS = 0.0f, banderaM = 0.0f, banderaH = 0.0f;
```

Para la animación del reloj se usaron banderas y variables para controlar las rotaciones y animación() que controla por máquina de estados cada uno de los ángulos a rotar en las manecillas

```
//animacion reloj
float rotKitS = 0.0f;
float rotKitM = 0.0f;
float rotKitH = 0.0f;
```

```
void animacion()
{
        if (limiteGrados < 360.0f)
        {
            //printf("limiteGrados:
            //printf("----------
            //printf("----------
            limiteGrados ++;
            //if (temp == 6.0f*10 )/
            if (temp == 6.0f)
            {
                banderaS=true;
                temp = 0.0f;
                //printf(banderaS ?
                //printf("----------
            }
            temp++;
            //tempS++;
            if (banderaS)
            {
                rotKitS += 6.0f;
```

Dentro del “main” se agregan los shaders y los modelos con sus respectivas ubicaciones

```
Shader lightingShader("Shaders/lighting.vs", "Shaders/lighting.frag");
Shader lampShader("Shaders/lamp.vs", "Shaders/lamp.frag");
Shader SkyBoxshader("Shaders/SkyBox.vs", "Shaders/SkyBox.frag");
Shader animShader("Shaders/anim.vs", "Shaders/anim.frag");

Model Piso((char*)"Models/pisoC/pisoC.obj");
//Model Esfera((char*)"Models/Esfera/Esfera.obj");
//Model Box((char*)"Models/Box/box.obj");
Model Cama((char*)"Models/cama/cama.obj");
Model silla((char*)"Models/silla/silla.obj");
Model sillon((char*)"Models/sillon/sillon.obj");
Model telefono((char*)"Models/telefono/telefono.obj");
Model muebles((char*)"Models/escritorio/muebles.obj");
Model impresora((char*)"Models/impresora/impresora.obj");
Model Girasol((char*)"Models/girasol/girasol.obj");
Model PastoFG((char*)"Models/pastoFG/pastoFG.obj");
Model plantaAP((char*)"Models/plantas/avedeParaiso.obj");
Model plantaAV((char*)"Models/plantas/aloeVera.obj");
Model plantaHC((char*)"Models/plantas/hojaCobre.obj");
Model plantaPS((char*)"Models/plantas/palmaSago.obj");
Model plantaP((char*)"Models/plantas/pasto.obj");
```

Es importante mantener los vertices del sky box para mantener un entorno agradable a la vista

```
GLfloat skyboxVertices[] = {
    // Positions
    -1.0f,  1.0f, -1.0f,
    -1.0f, -1.0f, -1.0f,
    1.0f, -1.0f, -1.0f,
    1.0f, -1.0f, -1.0f,
    1.0f,  1.0f, -1.0f,
    -1.0f,  1.0f, -1.0f,

    -1.0f, -1.0f,  1.0f,
    -1.0f, -1.0f, -1.0f,
    -1.0f,  1.0f, -1.0f,
    -1.0f,  1.0f, -1.0f,
    -1.0f,  1.0f,  1.0f,
```

```
//SkyBox
GLuint skyboxVBO, skyboxVAO;
glGenVertexArrays(1, &skyboxVAO);
glGenBuffers(1, &skyboxVBO);
glBindVertexArray(skyboxVAO);
glBindBuffer(GL_ARRAY_BUFFER, skyboxVBO
glBufferData(GL_ARRAY_BUFFER, sizeof(sk
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT, G

// Load textures
vector<const GLchar*> faces;
faces.push_back("SkyBox/right.tga");
faces.push_back("SkyBox/left.tga");
faces.push_back("SkyBox/top.tga");
faces.push_back("SkyBox/bottom.tga");
faces.push_back("SkyBox/back.tga");
```

En game loop se llaman las animaciones(), eventos(), texturas(), de esa manera se obtendra dentro de la ventana especifica los frames para visualizar el entorno.

Además, se usan los shaders para controlar iluminacion, texturas y animaciones, si asi se requiere por cada modelo, Pointlights, spootlight, etc. De esa manera se puede tener un control adecuado para cada una de las actividades que se quiera que realice un objeto como, iluminar o no, tener transparencia, incluso interactuar con el ususario por medios externos.

```
// 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 pr
    glfwPollEvents();
    DoMovement();
    animacion();

    // Clear the colorbuffer
    glClearColor(0.1f, 0.1f, 0.1f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // OpenGL options    revisar si quitar o no!!!!
    glEnable(GL_DEPTH_TEST);
```

```
glBindVertexArray(VAO);
glm::mat4 tmp = glm::mat4(1.0f); //Temp


glm::mat4 model(1);



//Carga de modelo
view = camera.GetViewMatrix();
model = glm::mat4(1);
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1i(glGetUniformLocation(lightingShader.Program, "activaTransparencia"), 0);
Piso.Draw(lightingShader);
```

Es importante limpiar memoria y recursos para mantener estable el uso de estas al finalizar el ciclo

```
glBindVertexArray(0);

// Draw skybox as last
glDepthFunc(GL_LEQUAL);  // Change depth function so
SkyBoxshader.Use();
view = glm::mat4(glm::mat3(camera.GetViewMatrix()));
glUniformMatrix4fv(glGetUniformLocation(SkyBoxshader.
glUniformMatrix4fv(glGetUniformLocation(SkyBoxshader.

// skybox cube
glBindVertexArray(skyboxVAO);
glActiveTexture(GL_TEXTURE1);
glBindTexture(GL_TEXTURE_CUBE_MAP, cubemapTexture);
glDrawArrays(GL_TRIANGLES, 0, 36);
glBindVertexArray(0);
glDepthFunc(GL_LESS); // Set depth function back to d


// Swap the screen buffers
glfwSwapBuffers(window);
```

Así como cerrar VAO, EVO, SKYBOX para terminar la ejecución del programa y evitar conflictos con el sistema operativo o aplicaciones.

```
glDeleteVertexArrays(1, &VAO);
glDeleteVertexArrays(1, &lightVAO);
glDeleteBuffers(1, &VBO);
glDeleteBuffers(1, &EBO);
glDeleteVertexArrays(1, &skyboxVAO);
glDeleteBuffers(1, &skyboxVBO);

// Terminate GLFW, clearing any resou
glfwTerminate();



return 0;
```

Conclusiones

Fue una de las mejores materias, teniendo complejidad, facilidad de materiales y software, sin duda, disfruté cada una de las prácticas, aunque incluso tuve problemas con algunas partes del modelado y animaciones, creo que comprendí bastante bien, espero mejorar un poco más en computación gráfica: sobre todo en parte de modelado jerárquico para sus animaciones por key frames.

Con respecto a la parte del proyecto, fue bastante complejo, desde crear texturas, ajustar los UVs, y sobre todo: animar, se requiere de mucha paciencia para hacer cada una de ellas, pero sin duda, disfrute muchísimo eso. Planeo seguir este campo, que es impresionante y un reto más, que merece la pena seguir.