TECHNICAL MANUAL

UNAM La Universidad de la Nación

Final Project

COMPUTER GRAPHICS LABORATORYING.

CARLOS ALDAIR ROMAN BALBUENA

Morales Esteban Irving

12 de mayo de 2022

Content

Objetive

The student must apply and demonstrate the knowledge acquired throughout the course.

The student must understand and understand the basic principles: basic transformations, texturing, and animation.

Gantt chart

Tasks

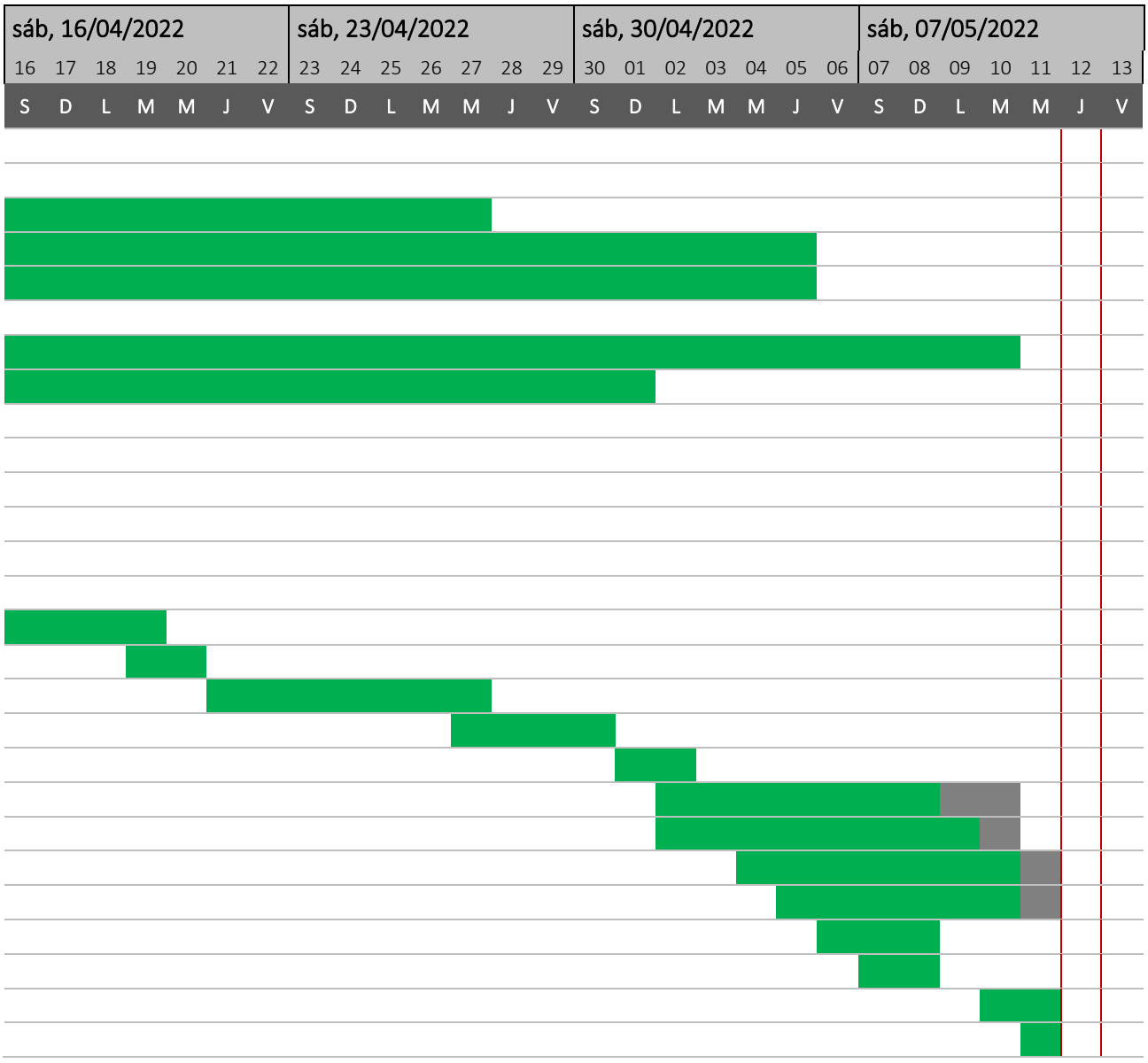
1. Learn how to use Git Hub
2. Learn how to use GIMP
3. Learn how to use MAYA
4. Learn to model in Open GL
5. Start with the technical manual
6. Start with the user manual
7. Feedback on handling pages you can use to download free models
8. Start loading the test repository
9. Upload the project repository
10. Model objects in the room including textures and colors
11. Model plants including texturing
12. Model walls including texturing.
13. Review and correct modeling errors.
14. Locate each object on the map.
15. Locate the walls on the map.
16. Review location errors.
17. Control animations.
18. Check that there are no animation problems.
19. Determine if extra elements are added and place them on the map.
20. Add such items if any.
21. Gather information about what was learned
22. Error feedback
23. Preliminary delivery of the project
24. Delivery of final project

FINAL Project

PROJECT LEADER: Morales Esteban Irving

			45	26/04/2022
		START OF THE PROJECT		sáb, 12/03/2022

TASK	RESPONSIBLE	PROGRESE	BEGINNING	END
PHASE 1				
Task 1	M. Irving	100%	20/03/2022	03/04/2022
Task 2	M. Irving	100%	20/03/2022	27/04/2022
Task 3	M. Irving	100%	12/03/2022	05/05/2022
Task 4	M. Irving	100%	12/03/2022	05/05/2022
PHASE 2	M. Irving			
Task 5	M. Irving	100%	23/03/2022	10/05/2022
Task 6	M. Irving	100%	23/03/2022	01/05/2022
Task 7	M. Irving	100%	24/03/2022	25/03/2022
PHASE 3	M. Irving			
Task 8	M. Irving	100%	24/03/2022	25/03/2022
Task 9	M. Irving	100%	25/03/2022	26/03/2022
Task 10	M. Irving	100%	26/03/2022	29/03/2022
Task 11	M. Irving	100%	29/03/2022	10/04/2022
Task 12	M. Irving	100%	08/04/2022	19/04/2022
Task 13	M. Irving	100%	19/04/2022	20/04/2022
Task 14	M. Irving	100%	21/04/2022	27/04/2022
Task 15	M. Irving	100%	27/04/2022	30/04/2022
Task 16	M. Irving	100%	01/05/2022	02/05/2022
Task 17	M. Irving	80%	02/05/2022	10/05/2022
Task 18	M. Irving	90%	02/05/2022	10/05/2022
Task 19	M. Irving	90%	04/05/2022	11/05/2022
Task 20	M. Irving	90%	05/05/2022	11/05/2022
Task 21	M. Irving	100%	06/05/2022	08/05/2022
Task 22	M. Irving	100%	07/05/2022	08/05/2022
Task 23	M. Irving	100%	10/05/2022	11/05/2022
Task 24	M. Irving	100%	11/05/2022	11/05/2022



Scope of the project

This project is for educational purposes, so it seeks to understand basic functionalities, therefore, small failures arise such as:

- Little visualization of the sky box inside the house room, through the windows, is not displayed,
- Transparency on crystals is only appreciated on objects, that is,
- chamber->crystal-> object.
- Transparency affects some objects that should not be transparent, it also depends on the type of material used on the object.
- You can only add textures on objects of type Lambert, Phong (color with png image, and specular).
- Some objects cannot be loaded properly (texture) if scaled with a 1000:2 ratio

Limiting

The limitations of the project are:

- Computer equipment: barely sustainable resources for the loading and execution of the project.
- Internet connection, project loading, slow
- Little optimization on objects by not maintaining a certain range of polygons on them.
- The scale is limited in the visual part of the camera, you cannot make an adequate approach on the objects.

Code documentation

```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"
```

The importance of each of the bookstores is vital to maintain order,

Each of them helps to:

• Use of window for the graphic environment

• Image display

• Use of arrays to contain the object

• Loading models by .obj files

• Camera functions

• Transformations on objects

• Texture loading from external media

• Bone animations

```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;
```

The prototype functions allow the handling of the graphic part by means of a window and through which you have a small visual field of the recreated environment, it is important to assign its dimensions.

The part of the camera is controlled by its initial position and variables that allow better control by means of the pointer, to simulate rotation, zoom and / or use of this by means of a preset control externally.

```
//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)
};
```

A spotlight to simulate an external lighting lamp

the initial points on the lighting are also established: position and direction

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

For the animation of the clock, flags and variables were used to control the rotations and animation () that controls by machine of states each of the angles to rotate in the hands

```
//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;
```

Inside the "main" shaders and models with their respective locations are added.

```
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");
```

It is important to maintain the vertices of the sky box to maintain a pleasant environment to the eye.

```
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");
```

In game loop are called the animations(), events(), textures(), that way you will get inside the window specifies the frames to visualize the environment.

In addition, shaders are used to control lighting, textures and animations, if required for each model, Pointlights, spootlight, etc. That way you can have an adequate control for each of the activities that you want an object to perform, such as, illuminate or not, have transparency, even interact with the user by external means.

```
// 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);
```

It is important to clean memory and resources to keep the use of these stable at the end of the cycle

```
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);
```

As well as closing VAO, EVO, SKYBOX to end the execution of the program and avoid conflicts with the operating system or applications.

```
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;
```

Conclusions

It was one of the best subjects, having complexity, ease of materials and software, without a doubt, I enjoyed each of the practices, although I even had problems with some parts of modeling and animations, I think I understood quite well, I hope to improve a little more in computer graphics: especially in part of hierarchical modeling for their animations by key frames.

Regarding the part of the project, it was quite complex, from creating textures, adjusting the UVs, and above all: animating, it takes a lot of patience to do each of them, but without a doubt, I enjoy that very much. I plan to follow this field, which is impressive and one more challenge, which is worth pursuing.