NATIONAL AUTONOMOUS UNIVERSITY OF MEXICO

FACULTY OF ENGINEERING

TECHNICAL MANUAL

Group: 12

Teacher:
Roman Balbuena Carlos Aldair

Student:
Mora Gonzalez Alan Francisco

Semester 2022-2
May 08, 2022

General index

Goal

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

Description

The student must select a facade and a space that can be real or fictitious and present reference images of said spaces for their 3D recreation in OpenGL.

In the reference image, 7 objects must be displayed that the student will recreate virtually and where said objects must be the closest thing to their reference image, as well as their setting.

A pdf document must be uploaded where it clearly shows its façade and the room to be recreated, as well as a list of the 7 objects that will be developed within it. This document must be uploaded before March 14.

Scopes

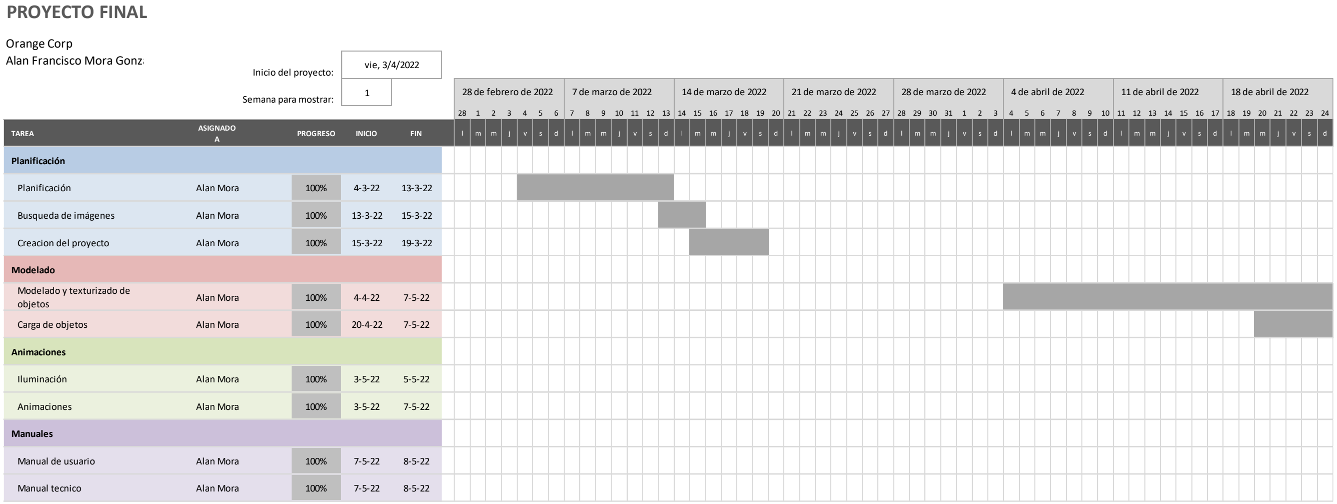
It seeks to recreate the animated space: McDonald's focusing on the exterior facade of the restaurant, as well as the living room together with part of the kitchen in OpenGL using 3d modeling to make the objects of the environment to be made with their respective texturing, this using the Maya Autodesk software to later load these 3D models in OpenGL using hierarchical modeling and geometric transformations. In addition, applying the knowledge of the graphic pipeline such as shaders, camera, drawing, projection, texturing, lighting, etc.

This space will have interaction with the user through the use of the camera to explore it, as well as simple and complex animations that offer a simple experience of immersion in the context for it.

Schedule of activities

Activities	Start	Duration (days)	End
Planning	March 4	9	March 13

Search for references	March 4	one	March 05
Project Creation	april 03	one	april 4
Modeling and texturing of objects	april 4	33	May 07
Loading of objects	April 20th	17	May 07
Lightning	May 03	two	May 05
animations	May 03	4	May 07
User manual	May 07	one	May 08
Technical manual	May 07	one	May 08



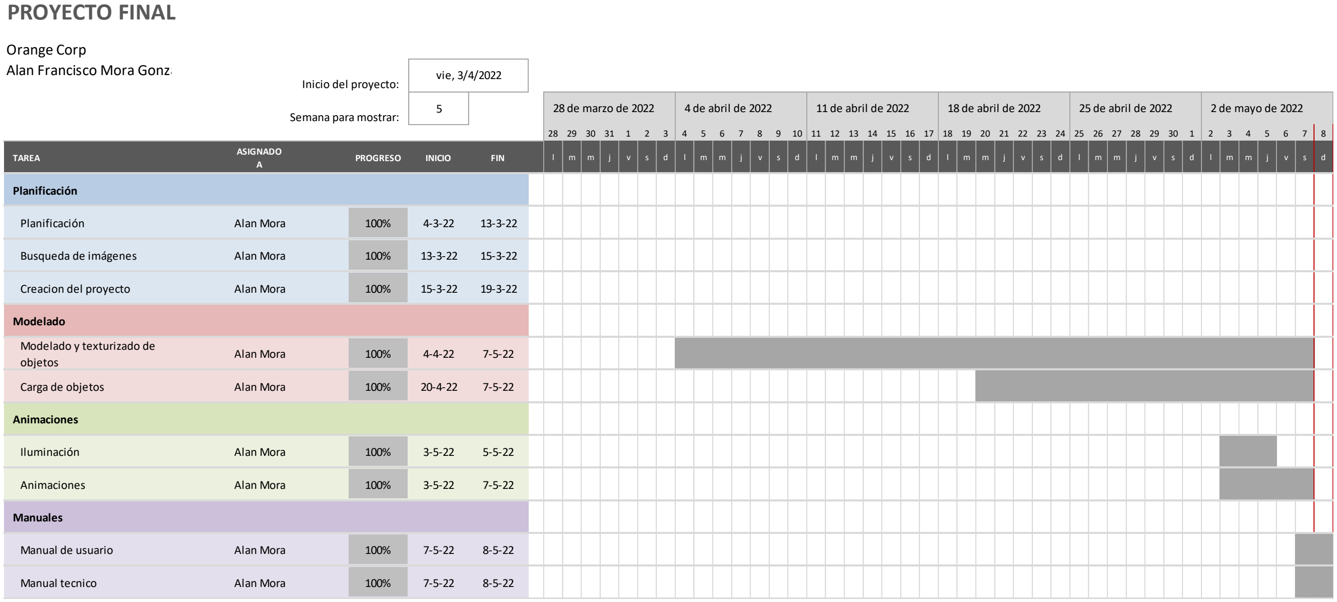
PROYECTO FINAL

Orange Corp
Alan Francisco Mora Gonz

Inicio del proyecto: vie, 3/4/2022

Semana para mostrar: 1

TAREA	ASIGNADO A	PROGRESO	INICIO	FIN
Planificación				
Planificación	Alan Mora	100%	4-3-22	13-3-22
Busqueda de imágenes	Alan Mora	100%	13-3-22	15-3-22
Creacion del proyecto	Alan Mora	100%	15-3-22	19-3-22
Modelado				
Modelado y texturizado de objetos	Alan Mora	100%	4-4-22	7-5-22
Carga de objetos	Alan Mora	100%	20-4-22	7-5-22
Animaciones				
Iluminación	Alan Mora	100%	3-5-22	5-5-22
Animaciones	Alan Mora	100%	3-5-22	7-5-22
Manuales				
Manual de usuario	Alan Mora	100%	7-5-22	8-5-22
Manual tecnico	Alan Mora	100%	7-5-22	8-5-22



PROYECTO FINAL

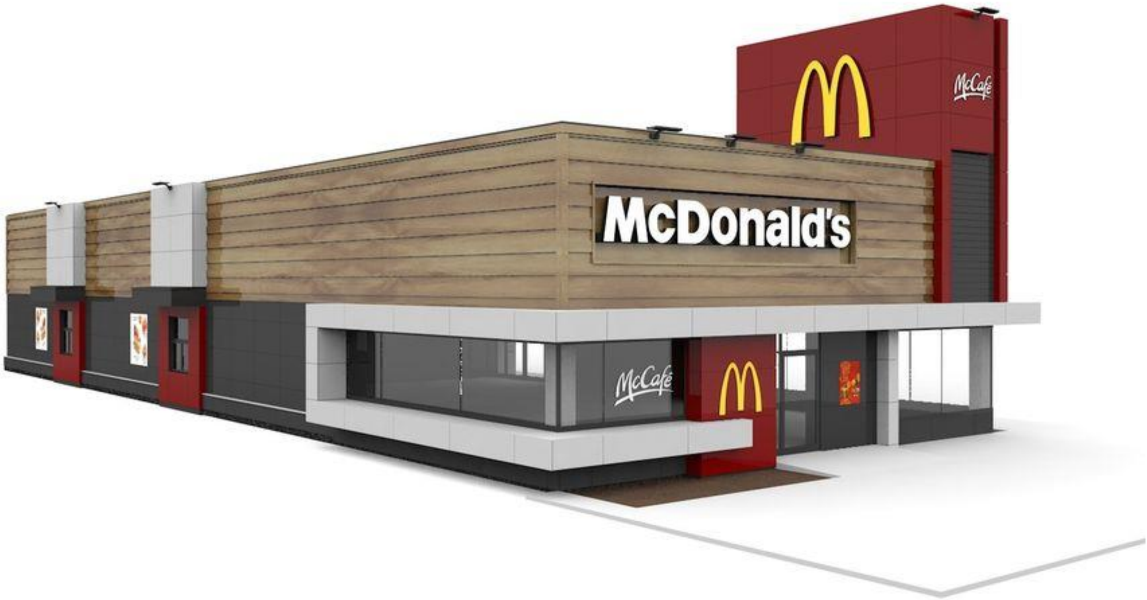
Orange Corp
Alan Francisco Mora Gonz

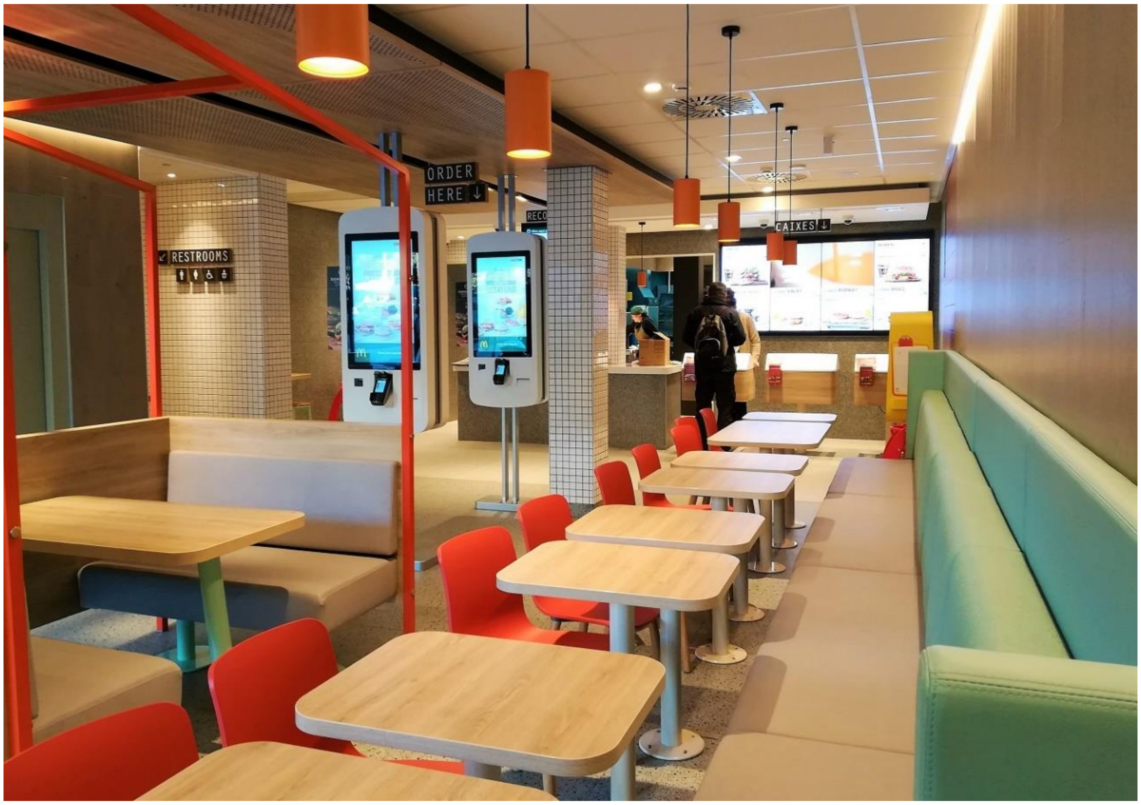
Inicio del proyecto: vie, 3/4/2022

Semana para mostrar: 5

TAREA	ASIGNADO A	PROGRESO	INICIO	FIN
Planificación				
Planificación	Alan Mora	100%	4-3-22	13-3-22
Busqueda de imágenes	Alan Mora	100%	13-3-22	15-3-22
Creacion del proyecto	Alan Mora	100%	15-3-22	19-3-22
Modelado				
Modelado y texturizado de objetos	Alan Mora	100%	4-4-22	7-5-22
Carga de objetos	Alan Mora	100%	20-4-22	7-5-22
Animaciones				
Iluminación	Alan Mora	100%	3-5-22	5-5-22
Animaciones	Alan Mora	100%	3-5-22	7-5-22
Manuales				
Manual de usuario	Alan Mora	100%	7-5-22	8-5-22
Manual tecnico	Alan Mora	100%	7-5-22	8-5-22

Reference images





Objects to recreate:

- Desk
- Chair
- Armchair
- self checkout machine
- Lamps
- Trash can
- order bar

Code

Libraries

```
// StdIncludes
#include<string>
#include<iostream>
#include<cmath>

// glew
#include<GL/glew.h>

// GLFW
#include<GLFW/glfw3.h>

// GL includes
#include"Shader.h"
#include"Camera.h"
#include"Model.h"

// GLM Mathematics
#include<glm/glm.hpp>
#include<glm/gtc/matrix_transform.hpp>
#include<glm/gtc/type_ptr.hpp>

// Other Libs
#include"SOIL2/SOIL2.h"
#include"stb_image.h"
```

External libraries

Library	Description
GLEW, GLFW	For communication with the OS and for handling windows in Visual Studio and OpenGL (creation, deletion, etc.).
GLM	C++ math library for projects on the OpenGL API.
Assimp	Library to import 3d models from programs like 3ds MAX, Maya, Blender, etc. and that OpenGL can understand its format, in our case it allows to import .obj models and read the .mtl files with the information of these objects to abstract to OpenGL.
SOIL2	Allows loading textures (images) in OpenGL.

Include files

Library	Description
Shader.h	It creates the vertex shader and fragment shader, all it receives is two addresses, the first for the vertex shader file and the second for the fragment shader file. It opens these files, recovering the code they contain and later creates both shaders assigning them memory locations, they are compiled, and it verifies that they have been created correctly.
Camera.h	Create the synthetic or interactive camera, this .h defines the movements that the camera will have (forward, backward, left and right), the positions or initial values of the camera for rotation, tilt, speed of movement, sensitivity and zoom or angle aperture for perspective projection.
Model.h	It receives the address of the .obj file and is responsible for importing said model using assimp, loading coordinates of its vertices (positions, normals and textures).

Shaders

shaders	Description
lamp.vs and lamp.frag	The vertex accesses the VAO memory location with the information about the position vertices and the arrays are affected with the position, the fragment displays the color information.
lighting.vs and lighting.frag	The vertex shader accesses all the VAO coordinates (position, color and texture), receives the model, view and projection matrices to modify them according to the geometry received from the VAO. The fragment shader defines the types of light that will be available (directional, point and spot) in the form of a structure with its components.

Functions

```
// Function prototypes
voidkeycallback(GLFWwindow*window,intkey,intscan code,intaction,intmode);
voidMouseCallback(GLFWwindow*window,doublexPos,doubleyPos);
voidDoMovement();
voidanimationChairs();
voidanimationBurger();
```

shaders	Description
keycallback	Processes keyboard input, the keys pressed to perform an action. 5 specific keys are considered for the interaction with the virtual environment [ESC, W, A, S, D]. Key events 0 to 1023 are also considered.
MouseCallback	It is the mouse input listener. The action you define for that input is the update on the camera rotation.
DoMovement	This function defines the actions to perform if there are activated animations started on KeyCallback, when they change state and also the camera update.
animationChairs	Within this animation is the necessary programming to do the complex animation of the restaurant chairs.
animationBurger	Inside this animation is the necessary programming to make the complex animation of the parabolic shot using a restaurant hamburger.
main	Main function with the program to set the virtual environment. Create the window, check for errors, call listener functions or callbacks for keyboard and mouse events. Defines the viewport, activates OpenGL options for handling translucent and/or transparent materials. Create and configure the shaders. Captures events from motion listeners and keyboard and mouse input. Define the lighting.

Global variables

Variable	Description
camera	Camera type with the initial position of the camera in space.
keys	Boolean array to handle keyboard input.
firstMouse	Boolean for camera handling.
WIDTH	Variable with the width of the window.
HEIGHT	Variable with the height of the window.
SCREEN_WIDTH and SCREEEN_HEIGHT	To store the width and height returned by some function.

transChair, rotChair and activeChair	Variables for simple animation of the chair.
transTrash, rotTrash and activeTras	Variables for simple trash can animation.
firstMouse	Boolean for camera handling.
movKitXChair, movKitYChair, and rotKitChair	Variables for complex animation of chairs.
circuitChair(1, 2)	Boolean variables to activate and deactivate the complex animation of the chairs, depending on these variables and the route that is activated will be the position of the chairs at that moment.
vinY, vinZ and rotKitBur	Variables for the complex animation of the hamburger.
burcircuit (1, 2)	Boolean variables to activate and deactivate the complex animation of the hamburger, depending on these variables and the route that is activated will be the position of the hamburger at that moment.
tour (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)	Boolean variables for the states of complex animations defined in KeyCallback.
lightPos	Illumination position attribute.
pointLightPositions	Fix with the coordinates of 5 point lights.
deltaTime	Time between the current frame and the last frame.
lastFrame	Time of the last frame.

Loading models (Objects).

Models are loaded through the creation of a Model Object for each model. Using Model.h specifying for each model the path of the .obj file so that the library or template is responsible for loading the models as explained in the include files section.

```
// Models
Modelchair((char*)"Models/Chair/chair.obj");
Modeltable((char*)"Models/Table/table.obj");
Modelcouch((char*)"Models/Sofa/sofa.obj");
Modelcounter((char*)"Models/Counter/counter.obj");
Modeltrash((char*)"Models/Trash/trash.obj");
Modeltop((char*)"Models/Trash/lid.obj");
Modelmachine((char*)"Models/Maquina/maquina.obj");
Modellamp((char*)"Models/Lamp/lamp.obj");
Modelburger((char*)"Models/Burger/burger.obj");
Modeltray((char*)"Models/Burger/tray.obj");

// Inside
Modelinside((char*)"Models/Interior/interior.obj");
Modelpillar((char*)"Models/Interior/pillar.obj");
```

```
//Exterior
Modelleft((char*)"Models/Restaurant/left.obj");
Modelright((char*)"Models/Restaurant/right.obj");
Modelceiling((char*)"Models/Restaurant/roof.obj");
```

Animations

The activation of the animations can be found in the KeyCallback function:

```
 if(keys[GLFW_KEY_R])
{
activeChair = !activeChair;
 if(activeChair)
{
transChair -= 1.0f;
rotChair += 50.0f;
}
 else
{
transChair += 1.0f;
rotChair -= 50.0f;
}
}

 if(keys[GLFW_KEY_F])
{
activeTrash = !activeTrash;
 if(activeTrash)
{
rotTrash += 60.0f;
transTrash += 2.2f;
}
 else
{
rotTrash -= 60.0f;
transTrash -= 2.2f;
}
}
```

```
 if(keys[GLFW_KEY_T])
{
active = !active;
 if(active)
{
Light1 = glm::vec3(0.0f, 0.0f, 1.0f);
Light2 = glm::vec3(0.0f, 0.0f, 1.0f);
Light3 = glm::vec3(0.0f, 0.0f, 1.0f);
Light4 = glm::vec3(0.0f, 0.0f, 1.0f);
Light5 = glm::vec3(0.0f, 0.0f, 1.0f);
}
 else
{
Light1 = glm::vec3(0);
Light2 = glm::vec3(0);
Light3 = glm::vec3(0);
Light4 = glm::vec3(0);
Light5 = glm::vec3(0);
}
}

 if(keys[GLFW_KEY_Z])
{
 if(circuitChair1)
{
circuitChair1 = false;
circuitChair2 = true;
}
 else
{
circuitChair1 = true;
circuitChair2 = false;
}
}
```

```
 if(keys[GLFW_KEY_X])
{
 if(circuitBur1)
{
circuitbur1 = false;
circuitbur2 = true;
}
 else
{
burcircuit1 = true;
circuitbur2 = false;
}
}
```

The first three animations consist of rotating and moving objects, the garbage cans and the chairs in the restaurant were animated. In addition, a light on and off was made as a simple animation.

The other two animations are complex, the movement of chairs makes transformations a little more complex than the simple ones, by measuring the rotations and translations necessary to simulate the lifting of the chairs. While the jump made by the hamburger makes use of the parabolic shot.