Universidad Nacional Autónoma de México

Facultad de Ingeniería

Ingeniería en Computación

Computación Gráfica e Interacción Humano-Computadora

PROYECTO FINAL

RESIDENCIA DE RED

Grupo 4

Alumno:

Luna Pérez José Luis

Prof. Ing. Carlos Aldair Román Balbuena

México, Ciudad de México. Martes 26 de enero, 2021

Objetivos

-El alumno realizara un escenario real o ficticio en modelado 3D con lo aprendido en el curso.

Propuesta

La propuesta se realizó a partir de la casa de red el protagonista de Pokémon Rojo fuego. Las imágenes referencia son:

El único error que se detectó con las imágenes referencia son las escaleras ya que es necesario modificar la fachada original, ya que como se observan en las imágenes las escaleras en la planta baja se encuentran en un extremo, mientras que en la parte superior se encuentra un tanto centrada.

Los objetos propuestos para modelar son:

En el cuarto los objetos que recrearán serán:

-La mesa

-Cajonera

-Cama

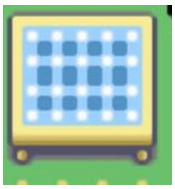
-Librero

-Televisor

El siguiente cuarto es el comedor:

-Silla:

-Mesa:



-Lavabo:

-Exhibidor:

- Televisor y bote de basura.

Diagrama de Gantt

	10 de diciembre de 2020	14 de diciembre de 2020	15 de diciembre de 2020	16 de diciembre de 2020	22 de enero de 2021	26 de enero de 2021
Entrega final de la propuesta de casa						
Inicio de la búsqueda de información de modelado						
Elección de 3ds como herramienta de modelado						
Se crea repositorio en GitHub al igual que el proyecto en visual						
Se empieza el modelado de los objetos.						
Se integra en proyecto los modelos.						
Se realizan entregables y últimos toques.						

Documentación

Primeramente, el inicio del código se definirán los encabezados que se usaran en el programa para generar funciones u objetos de utilidad.

```
#include <Windows.h>
#include <glad/glad.h>
#include <glfw3.h>  //main
#include <stdlib.h>
#include <glm/glm.hpp>  //camera y model
#include <glm/gtc/matrix_transform.hpp> //camera y model
#include <glm/gtc/type_ptr.hpp>
#include<time.h>


#define STB_IMAGE_IMPLEMENTATION
#include <stb_image.h>  //Texture

#define SDL_MAIN_HANDLED
#include <SDL/SDL.h>

#include <shader_m.h>
#include <camera.h>
#include <modelAnim.h>
#include <model.h>
#include <Skybox.h>
#include <iostream>
```

A continuación, se declararan las funciones, se configura el tamaño de pantalla que será una resolución de 800*600 y configuramos la cámara para poder usarla de manera satisfactoria en el proyecto.

```
//#pragma comment(lib, "winmm.lib")

void framebuffer_size_callback(GLFWwindow* window, int width, int height);
void mouse_callback(GLFWwindow* window, double xpos, double ypos);
void scroll_callback(GLFWwindow* window, double xoffset, double yoffset);
void my_input(GLFWwindow *window);
void animate(void);

// settings
const unsigned int SCR_WIDTH = 800;
const unsigned int SCR_HEIGHT = 600;

// camera
Camera camera(glm::vec3(0.0f, 0.0f, 3.0f));
float MovementSpeed = 0.1f;
float lastX = SCR_WIDTH / 2.0f;
float lastY = SCR_HEIGHT / 2.0f;
bool firstMouse = true;
```

Se considera el tiempo de frames que se usara y además agregamos 3 luces que nos ayudaran a iluminar el escenario. Primeramente, le daremos la posición deseada de la luz la cual se obtuvo al conocer la posición de la cámara y después la dirección en la que los rayos se propagaran.

```
// timing
const int FPS = 60;
const int LOOP_TIME = 1000 / FPS; // = 16 milisec // 1000 millisec == 1 sec
double  deltaTime = 0.0f,
        lastFrame = 0.0f;

// Light
//glm::vec3 lightDirection = glm::vec3(-0.2f, -1.0f, -0.3f);
//Lighting
glm::vec3 lightPosition(0.0f, 4.0f, -10.0f);
glm::vec3 lightDirection(-1.0f, -1.0f, -1.0f);

glm::vec3 lightPosition2(0.0f, -1.5f, -1.0f);
glm::vec3 lightDirection2(0.0f, -1.0f, 0.0f);

glm::vec3 lightPosition3(0.0f, 0.0f, -10.0f);
glm::vec3 lightDirection3(-1.0f, -1.0f, -1.0f);
```

En esta función que se encuentra en main la cual se encarga de crear la ventana.

```
GLFWwindow* window = glfwCreateWindow(SCR_WIDTH, SCR_HEIGHT, "Pokemon Red", NULL, NULL);
if (window == NULL)
{
    std::cout << "Failed to create GLFW window" << std::endl;
    glfwTerminate();
    return -1;
}
```

Para crear el skybox usaremos el siguiente segmento:

```
//Shader staticShader("Shaders/lightVertex.vs", "Shaders/lightFragment.fs");
Shader staticShader("Shaders/shader_Lights.vs", "Shaders/shader_Lights.fs");
Shader skyboxShader("Shaders/skybox.vs", "Shaders/skybox.fs");

vector<std::string> faces
{
    "resources/skybox/right.jpg",
    "resources/skybox/left.jpg",
    "resources/skybox/top.jpg",
    "resources/skybox/bottom.jpg",
    "resources/skybox/front.jpg",
    "resources/skybox/back.jpg"
};

Skybox skybox = Skybox(faces);

// Shader configuration
// --------------------
skyboxShader.use();
skyboxShader.setInt("skybox", 0);
```

En donde cargamos los shaders y cargamos el recurso en imagen que usaremos para texturizar el skybox.

Aquí se declaran las instancias de los modelos que se usaran en el programa.

```
Model piso("resources/objects/pisof/pasto.obj");
Model buzon("resources/objects/Buzon/buzon.obj");
Model plantas("resources/objects/planta/planta.obj");
Model cuadro("resources/objects/cuadro/cuadro.obj");
Model cuadro2("resources/objects/cuadro2/cuadro.obj");
Model cuarto1("resources/objects/CuartoT/cuarto.obj");
Model mesa("resources/objects/Mesa/mesa.obj");
Model estante("resources/objects/estan_a/estante.obj");
Model basura("resources/objects/basura/basura.obj");
Model tele("resources/objects/tele/tele.obj");
Model tele2("resources/objects/tele/tele.obj");
Model lava("resources/objects/lava/lavabo.obj");
Model poke("resources/objects/poke/poke.obj");
Model casar("resources/objects/CasaR/casa.obj");
Model techo("resources/objects/techo/techo.obj");
Model techo2("resources/objects/techo2/techo.obj");
```

Para modificar el objeto primero modificamos donde se colocará la figura con las transformaciones básicas y nombrando el modelo que se usará.

```
model = glm::mat4(1.0f);
model = glm::rotate(glm::mat4(1.0f), glm::radians(90.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::translate(model, glm::vec3(15.0f, -1.5f, 0.0f));
model = glm::scale(model, glm::vec3(0.01f, 0.01f, 0.01f));
staticShader.setMat4("model", model);
plantas.Draw(staticShader);
```

La cámara será desplazada con las letras w (frente), s (atrás), a(derecha) y d(izquierda). También se incorpora dos teclas que tendrán una importancia relevante que sería el escape que cerrara la ventana y el espacio muestra la posición en coordenadas de la cámara.

```
if (glfwGetKey(window, GLFW_KEY_ESCAPE) == GLFW_PRESS)
    glfwSetWindowShouldClose(window, true);

if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS)
    camera.ProcessKeyboard(FORWARD, deltaTime);
if (glfwGetKey(window, GLFW_KEY_S) == GLFW_PRESS)
    camera.ProcessKeyboard(BACKWARD, deltaTime);
if (glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS)
    camera.ProcessKeyboard(LEFT, deltaTime);
if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS)
    camera.ProcessKeyboard(RIGHT, deltaTime);
if (glfwGetKey(window, GLFW_KEY_SPACE) == GLFW_PRESS)
    animacion = true;
```

Este segmento modifica la velocidad de la cámara al presionar el shift.

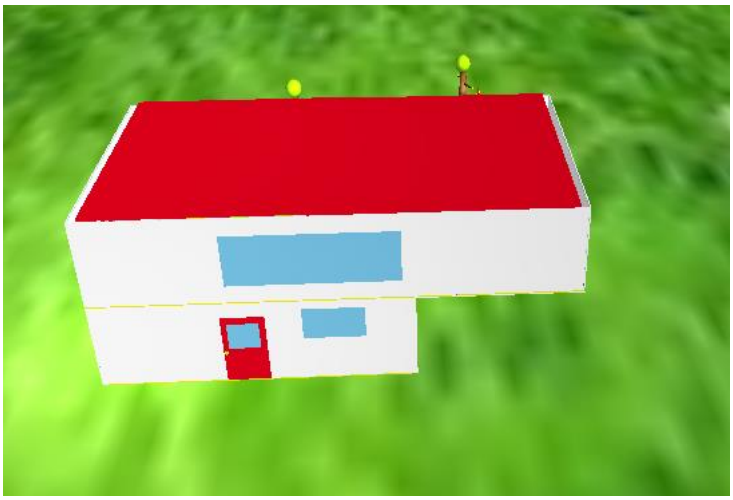
```
if (glfwGetKey(window, GLFW_KEY_LEFT_SHIFT) == GLFW_PRESS)
    camera.MovementSpeed = MovementSpeed * 2.5f;
else
    camera.MovementSpeed = MovementSpeed;
```

Animación

```
if (glfwGetKey(window, GLFW_KEY_X) == GLFW_PRESS) {
    if (!tec)
        tec = true;
    else
        tec = false;
    fflush(stdin);
    std::cout << "tec=" << tec<< std::endl;
}

if (glfwGetKey(window, GLFW_KEY_Z) == GLFW_PRESS) {
    if (!tec2)
        tec2 = true;
    else
        tec2 = false;
    fflush(stdin);
    std::cout << "tec2=" << tec2 << std::endl;
}
```

El código de la animación del techo se debe presionar la tecla x y z lo cual se recorre el techo para poder observar lo que hay en la parte superior de la casa. Solo se recorrerá cuando se presiona la tecla para activar las banderas.



Costos

Tiempo				
Horas aprox	35			
$ por hora	200			
Hora total $	7000			
Servicios				
Luz	300		Total	7500
Agua	200		Total+iva	8700
Conocimientos				
Tutoriales youtube	0			
Software	0			
Clases Fi	0			
Iva	0.16	1200		