UNIVERSIDAD NACIONAL
AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

MANUAL TÉCNICO
PROYECTO FINAL

COMPUTACION GRÁFICA E INTERACCION HUMANO COMPUTADORA

315114078

GRUPO: 09

SEMESTRE 2022-2

Contenido

Objetivo

Con el propósito de implementar los conocimientos aprendidos durante el curso de *computación grafica e interacción humano computadora* se desarrolló el presente proyecto final . Así también este proyecto tiene como finalidad el poder crear un entorno virtual recreado de una facha y un cuarto, en donde existan animaciones sencillas y complejas elaboradas por el programador y con las cuales el usuario entre en un ambiente inmersivo en donde se sienta parte del entorno virtual y pueda manipular las animaciones.

Alcance

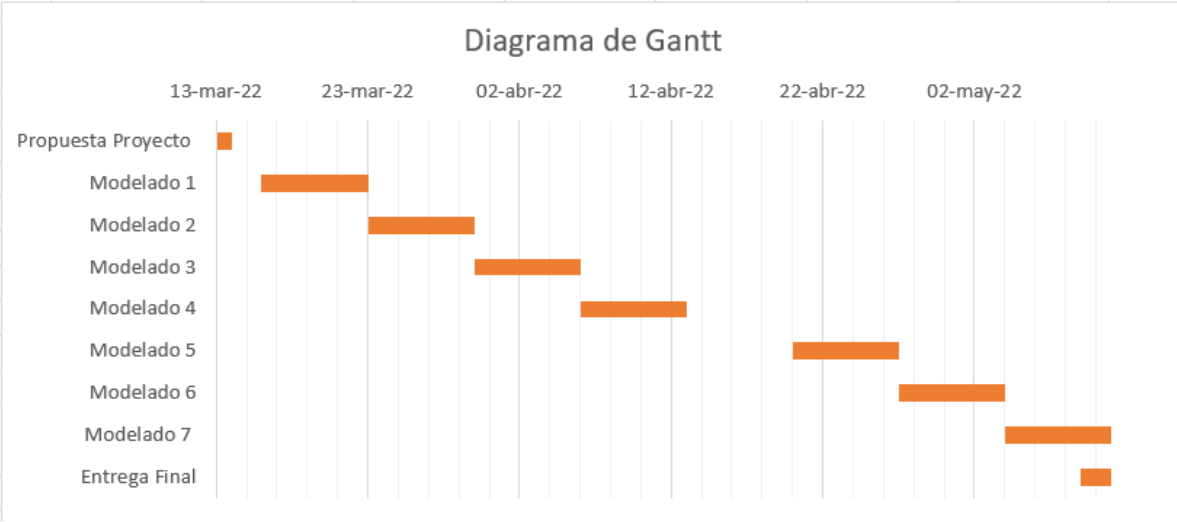
El alcance de este proyecto es poder llegar a un programa desarrollado en OpenGL y Maya que represente y se parezca a un entro real. Así como llegar a implantar todos los conocimientos previos aprendidos en clase como lo son la creación de geometrías, aplicación de texturas, la interacción con la cámara del entorno y las animaciones simples y complejas. Con todo ellos tiene cono alcance desarrollar un entorno virtual que sea ejecutable en cualquier aparato de computo y que funcione correctamente.

Limitaciones

Para el desarrollo de este proyecto se tuvo como limitaciones para la elaboración del proyecto el equipo de hardware utilizado pues no contaba con la potencia se necesitaba para ejecutar los softwares necesarios para la realización del proyecto.

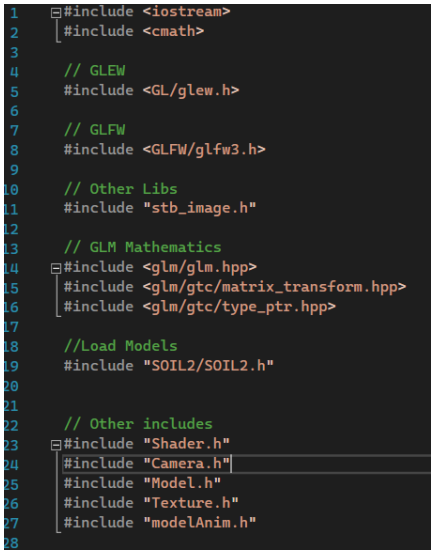
También se vio como limitante el software Maya utilizado para la creación de modelos ya que no funcionaba de la manera más optima, lo que llevo a que el proyecto se atrasara un poco.

Diagrama de Gantt



Documentación

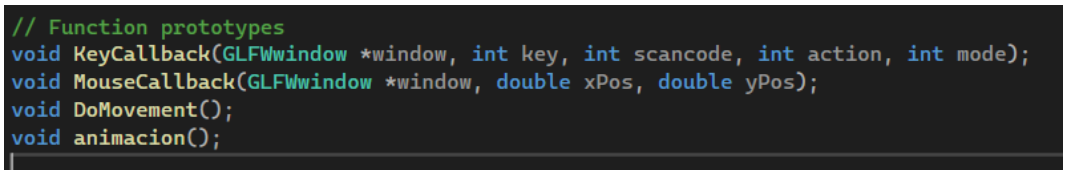
- Librerías
  En esta parte se incluyen todas la librerías que se utilizan en el programa, como lo son: librerías de inclusión de código en OpenGL, para realizar transformaciones a gml, etc.



```
#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"
```

Ilustración 1 Librerias

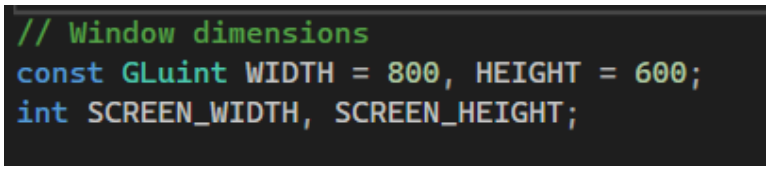
- Declaración de funciones
  Aquí se declaran las funciones del manejo de teclado y mouse, así como las funciones para las animaciones.



```
// 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();
```

Ilustración 2 Funciones

- Venta de visualización
  En esta parte se determina el tamaño de la pantalla que se vera al ejecutar el programa.



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

Ilustración 3 Ventana de visualización

- Posición cámara
  Se designan los valores en los que estará la cámara, por lo general en el punto media de la venta de visualización.

```
// Camera
Camera  camera(glm::vec3(-95.0f, 6.0f, -10.0f));
GLfloat lastX = WIDTH / 2.0;
GLfloat lastY = HEIGHT / 2.0;
bool keys[1024];
bool firstMouse = true;
float range = 0.0f;
float rot = 0.0f;
float movCamera = 0.0f;
```

Ilustración 4 Posición cámara

- Variables de animaciones básicas
  Aquí se declaran las variables de activación de animaciones e incremento o decremento de animaciones básicas

```
// Animaciones
float trans1 = 0.0f;
float trans2 = 0.0f;
float trans3 = 0.0f;
float rot1 = 0.0f;
float rotG = 0.0f;
bool animG1 = false;
bool animG2 = true;
bool animG3 = false;
bool anim1T= false;
bool anim1F = false;
bool anim2T = false;
bool anim2F = false;
bool anim3T = false;
bool anim3F = false;
bool anim4T = false;
bool anim4F = false;
```

Ilustración 5 Variables de animaciones básicas

- Atributos de luz
  Define las posiciones iniciales de las luces y hacia donde apuntan, en especifico de la luz direccional

```
// Light attributes
glm::vec3 lightPos(0.0f, 0.0f, 0.0f);
glm::vec3 PosIni(-95.0f, 1.0f, -45.0f);
glm::vec3 PosIni2(-95.0f, 1.0f, -45.0f);
glm::vec3 lightDirection(0.0f, -1.0f, -1.0f);
```

Ilustración 6 Atributos de luz

- Puntos de luz
  Se definen que punto de luz ocupar y las coordenadas de donde estará dicha punto de luz, en este caso fue en la posición de la lampara.

```
// Positions of the point lights
glm::vec3 pointLightPositions[] = {
    glm::vec3(posX-0.276,posY+2.931,posZ-5.880),
    glm::vec3(0,0,0),
    glm::vec3(0,0,0),
    glm::vec3(0,0,0)
};
```

Ilustración 7 Punto de luz

- Variables de animaciones complejas
  Aquí se declaran las variables de activación de animaciones e incremento o decremento de animaciones complejas

```
//Animación del gato
float movKitX = 0.0;
float movKitZ = 0.0;
float rotKit = 0.0;

bool circuito = false;
bool recorrido1 = true;
bool recorrido2 = false;
bool recorrido3 = false;
bool recorrido4 = false;
bool recorrido5 = false;

//Animación del coche
float movKitCX = 0.0;
float movKitCZ = 0.0;
float rotKitC = 0.0;

bool circuitoC = false;
bool recorridoC1 = true;
bool recorridoC2 = false;
bool recorridoC3 = false;
bool recorridoC4 = false;
bool recorridoC5 = false;
bool recorridoC6 = false;
```

Ilustración 8 Variables de animación compleja

- Declaración de modelos
  Se declaran los modelos que se van a cargar en el programa.

```
Model Piso((char*)"Models/Piso/Piso.obj");
Model Fachada((char*)"Models/Exterior/Fachada.obj");
Model PuertaIzq((char*)"Models/Exterior/PuertaIzq.obj");
Model PuertaDer((char*)"Models/Exterior/PuertaDer.obj");
Model Cama((char*)"Models/Cuarto/Cama.obj");
Model Buro((char*)"Models/Cuarto/Buro.obj");
Model Lampara((char*)"Models/Cuarto/Lampara.obj");
Model Escritorio((char*)"Models/Cuarto/Escritorio.obj");
Model Estante((char*)"Models/Cuarto/Estante.obj");
Model Librero((char*)"Models/Cuarto/Librero.obj");
Model CajasArriba((char*)"Models/Cuarto/CajasArriba.obj");
Model CajaAbajo((char*)"Models/Cuarto/CajaAbajo.obj");
Model Libros((char*)"Models/Cuarto/Libros.obj");
Model Silla((char*)"Models/Cuarto/Silla.obj");
Model Coche((char*)"Models/Coche/Coche.obj");
Model CuerpoGato((char*)"Models/Gato2/CuerpoGato.obj");
Model PataDelIzq((char*)"Models/Gato2/PataDelIzq.obj");
Model PataDelDer((char*)"Models/Gato2/PataDelDer.obj");
Model PataTrasIzq((char*)"Models/Gato2/PataTrasIzq.obj");
Model PataTrasDer((char*)"Models/Gato2/PataTrasDer.obj");
```

Ilustración 9 Modelos

- Vértices del skybox
  Se define que tamaño tendrá el cubo del skybox

```
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,
    -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,
    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,
    1.0f, -1.0f, -1.0f,
    -1.0f, -1.0f,  1.0f,
    1.0f, -1.0f,  1.0f
};
```

Ilustración 10 Vertices skybox

- Skybox
  Se define el skybox y cuales serán las texturas que tendrá cada parte de él.

```
//SkyBox
GLuint skyboxVBO, skyboxVAO;
glGenVertexArrays(1, &skyboxVAO);
glGenBuffers(1,&skyboxVBO);
glBindVertexArray(skyboxVAO);
glBindBuffer(GL_ARRAY_BUFFER, skyboxVBO);
glBufferData(GL_ARRAY_BUFFER, sizeof(skyboxVertices),&skyboxVertices,GL_STATIC_DRAW);
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT,GL_FALSE, 3 * sizeof(GLfloat), (GLvoid *)0);

// Load textures
vector<const GLchar*> faces;
faces.push_back("SkyBox/SkyBox_right.tga");
faces.push_back("SkyBox/SkyBox_left.tga");
faces.push_back("SkyBox/SkyBox_top.tga");
faces.push_back("SkyBox/SkyBox_bottom.tga");
faces.push_back("SkyBox/SkyBox_back.tga");
faces.push_back("SkyBox/SkyBox_front.tga");
```

Ilustración 11 Skybox

- Puntos de luces
  Se establecen las propiedades de iluminación difusa, especular, ambienta. Así como coleres que proyectarán y la intensidad con la que lo harán.

```
// Point light 1
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].position"), pointLightPositions[0].x, pointLightPositions[0].y, pointLightPositions[0].z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].ambient"), 0.05f, 0.05f, 0.05f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].diffuse"), LightP1.x, LightP1.y, LightP1.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].specular"), LightP1.x, LightP1.y, LightP1.z);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].linear"), 0.1f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].quadratic"), 1.0f);
```

Ilustración 12 Punto de Luz

- Carga de modelos
  En esta parte del código se cargan todos los modelos que se quieran dibujar en el entorno, indicando sus translaciones y rotaciones particulares de cada modelo a cargar. También en esta parte se mandan a dibujar los modelos.

```
  //Carga de modelo

  view = camera.GetViewMatrix();
  glm::mat4 model(1);
  tmp = model = glm::translate(model, glm::vec3(0, 1, 0));
  model = glm::translate(model,glm::vec3(posX,posY,posZ));
  model = glm::rotate(model, glm::radians(rot), glm::vec3(0.0f, 1.0f, 0.0));
  glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
  Piso.Draw(lightingShader);


  //Fachada
  view = camera.GetViewMatrix();
  model = glm::translate(tmp, glm::vec3(0.0f, 0.0f, 0.0f));
  model = glm::translate(model, glm::vec3(posX, posY, posZ));
  model = glm::rotate(model, glm::radians(rot), glm::vec3(0.0f, 1.0f, 0.0));
  glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
  glUniform1f(glGetUniformLocation(lightingShader.Program, "transparencia"), 0.0);
  Fachada.Draw(lightingShader);
///    Puerta Izq
  view = camera.GetViewMatrix();
  model = glm::translate(tmp, glm::vec3(-1.927f, 2.520f, 6.373f));
  model = glm::translate(model, glm::vec3(posX, posY, posZ));
  model = glm::rotate(model, glm::radians(rot1), glm::vec3(0.0f, 1.0f, 0.0));
  glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
  glUniform1f(glGetUniformLocation(lightingShader.Program, "transparencia"), 0.0);
  PuertaIzq.Draw(lightingShader);
```

Ilustración 13 Carga de modelos 1

```
//      Puerta Der
        view = camera.GetViewMatrix();
        model = glm::translate(tmp, glm::vec3(1.892f, 2.520f, 6.373f));
        model = glm::translate(model, glm::vec3(posX, posY, posZ));
        model = glm::rotate(model, glm::radians(-rot1), glm::vec3(0.0f, 1.0f, 0.0));
        glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
        glUniform1f(glGetUniformLocation(lightingShader.Program, "transparencia"), 0.0);
        PuertaDer.Draw(lightingShader);

//      Cama
        view = camera.GetViewMatrix();
        model = glm::translate(tmp, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(posX, posY, posZ));
        model = glm::rotate(model, glm::radians(rot), glm::vec3(0.0f, 1.0f, 0.0));
        glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
        glUniform1f(glGetUniformLocation(lightingShader.Program, "transparencia"), 0.0);
        Cama.Draw(lightingShader);
//      Buro
        view = camera.GetViewMatrix();
        model = glm::translate(tmp, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(posX, posY, posZ));
        model = glm::rotate(model, glm::radians(rot), glm::vec3(0.0f, 1.0f, 0.0));
        glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
        glUniform1f(glGetUniformLocation(lightingShader.Program, "transparencia"), 0.0);
        Buro.Draw(lightingShader);
//      Lampara
        view = camera.GetViewMatrix();
        model = glm::translate(tmp, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(posX, posY, posZ));
        model = glm::rotate(model, glm::radians(rot), glm::vec3(0.0f, 1.0f, 0.0));
        glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
        glUniform1f(glGetUniformLocation(lightingShader.Program, "transparencia"), 0.0);
        Lampara.Draw(lightingShader);
```

Ilustración 14 Carga de modelos 2

- Animaciones

  Parte del código en donde de define que aminación hará cada modelo, aquí se define que recorridos hará el modelo, cuantos grados debe girar y a que velocidad ira.

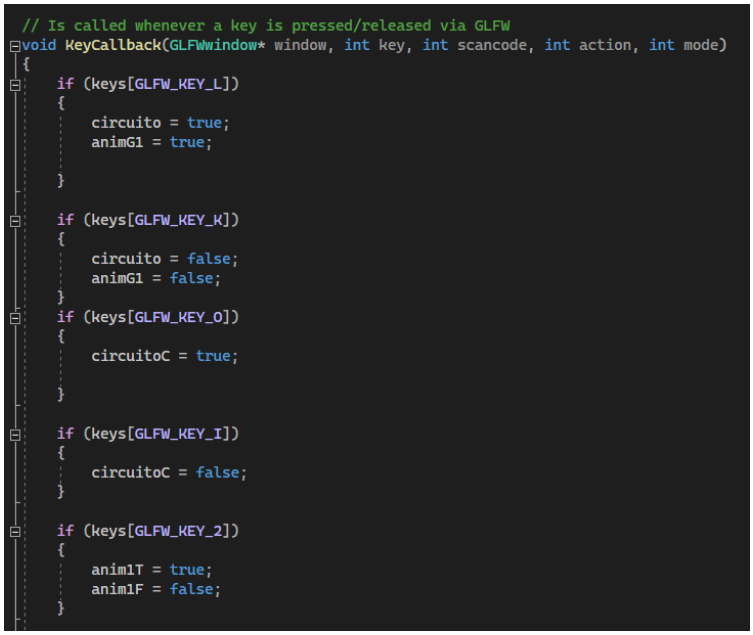
```
void animacion()
{
    //Movimiento del gato
    if (circuito)
    {
        if (recorrido1)
        {
            movKitZ += 0.005f;
            if (movKitZ > 3)
            {
                recorrido1 = false;
                recorrido2 = true;
            }
        }
        if (recorrido2)
        {
            rotKit = 90;
            movKitX += 0.005f;
            if (movKitX > 2)
            {
                recorrido2 = false;
                recorrido3 = true;

            }
        }
```

I lustración 16 animacion gato1

```
if (recorrido3)
{
    rotKit = 215;
    movKitZ -= 0.005f;
    movKitX -= 0.005f;
    if (movKitX < -2)
    {
        recorrido3 = false;
        recorrido4 = true;
    }
}

if (recorrido4)
{
    rotKit = 90;
    movKitX += 0.005f;
    if (movKitX > 0)
    {
        recorrido4 = false;
        recorrido5 = true;
    }
}
if (recorrido5)
{
    rotKit = 0;
    movKitZ += 0.01f;
    if (movKitZ > 0)
    {
        recorrido5 = false;
        recorrido1 = true;
    }
}
```

Ilustración 15 animacion gato 2

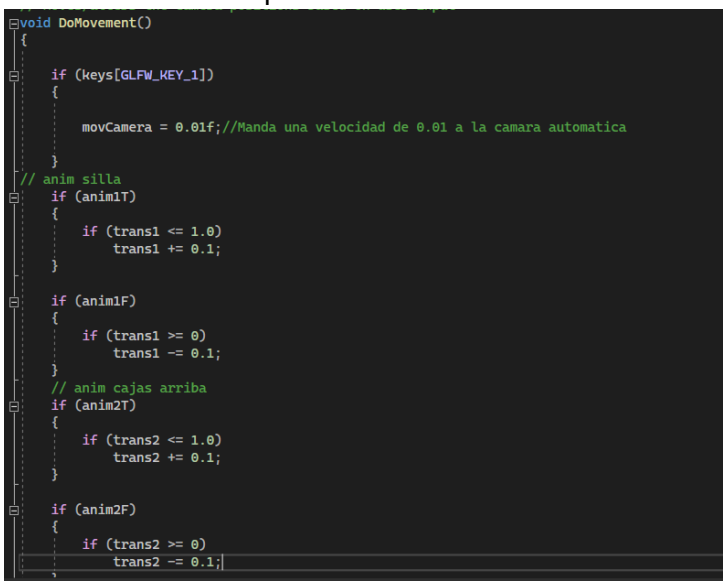
- Asignación de teclas
  Se asignan dientes teclas para cada uno de los movimientos de las animaciones que declararon anteriormente, así como para el manejo de la cámara.



```
// Is called whenever a key is pressed/released via GLFW
void KeyCallback(GLFWwindow* window, int key, int scancode, int action, int mode)
{
    if (keys[GLFW_KEY_L])
    {
        circuito = true;
        animG1 = true;

    }

    if (keys[GLFW_KEY_K])
    {
        circuito = false;
        animG1 = false;
    }
    if (keys[GLFW_KEY_O])
    {
        circuitoC = true;

    }

    if (keys[GLFW_KEY_I])
    {
        circuitoC = false;
    }

    if (keys[GLFW_KEY_2])
    {
        anim1T = true;
        anim1F = false;
    }
```

Ilustración 17 Asignación teclas

- DoMovement
  Se define los limites de las animaciones, como se van a mover reconociendo cuando se activa la animación o cuando se pausa.



```
void DoMovement()
{

    if (keys[GLFW_KEY_1])
    {

        movCamera = 0.01f;//Manda una velocidad de 0.01 a la camara automatica

    }
// anim silla
    if (anim1T)
    {
        if (trans1 <= 1.0)
            trans1 += 0.1;
    }

    if (anim1F)
    {
        if (trans1 >= 0)
            trans1 -= 0.1;
    }
    // anim cajas arriba
    if (anim2T)
    {
        if (trans2 <= 1.0)
            trans2 += 0.1;
    }

    if (anim2F)
    {
        if (trans2 >= 0)
            trans2 -= 0.1;
```

Ilustración 18 DoMovement

Conclusiones

Con este proyecto se pudieron aplicar los conocimientos vistos en clase, gracias que conforme avanzamos en temas íbamos avanzado con el proyecto se pudo poner en práctica cada uno de los temas con una aplicación bastante detallada a mi parecer. Para la finalización del proyecto estuvo interesante que cada quien pudiera poner parte de su creatividad a su proyecto tanto en lo de diseño como en la programación, pues se nos dio la libertad de elegir y jugar con las animaciones que nos gustaran más, esto hizo que los conocimientos que requería el proyecto quedaran aún mejor entendidos.

Por otro lado, me gusto bastante que se pudo aprender a manejar dos softwares totalmente diferentes pero que se complementan a la vez. La implementación de estas dos herramientas de trabajo nos amplia de gran manera nuestros conocimientos, lo cual se agradece mucho.