

# **UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

## **FACULTAD DE INGENIERÍA**



**ING. CARLOS ALDAIR ROMAN BALBUENA**  
**(LAB. COMPUTACION GRAFICA E INTERACCION**  
**HUMANO-COMPUTADORA)**

**Manual Técnico.**

**Semestre:2021-2**  
**Grupo: 09**

**Alumno:**

- Guerra Silva Erick Ivan

**Fecha de entrega: 28-06-2021**



<b>Objetivos.</b>	<b>3</b>
<b>Alcance del proyecto.</b>	<b>3</b>
<b>Limitantes.</b>	<b>3</b>
<b>Diagrama de Gantt.</b>	<b>4</b>
<b>Funcionamiento del proyecto.</b>	<b>15</b>
<b>Conclusión.</b>	<b>17</b>

## Objetivos.

El objetivo es recrear una fachada con diferentes objetos, esto con la finalidad de demostrar los conocimientos adquiridos en el laboratorio de la materia de Computación Gráfica e Interacción Humano-Computadora.

## Alcance del proyecto.

Recreación de la casa de “Timmy Turner”, esto a partir del modelado en Maya y OpenGL, sin embargo se ambientara su cuarto, ya que en ese lugar se importarán los objetos que en este caso son los siguientes.

- ❖ Cama
- ❖ Cuadro
- ❖ Pecera
- ❖ Buro de pecera
- ❖ Silla de pc
- ❖ Escritorio
- ❖ Pc

Por otra parte también se incluirá el personaje de Timmy Turner en su habitación, un varita en el primer piso y Timmy Turner superhéroe en la entrada de la casa, estos últimos para poder realizar animaciones por keyframes.

Otras partes importantes de este proyecto son:

- ☐ 5 animaciones 3 sencillas y 2 complejas.
- ☐ Manejo de la cámara para recorrer el ambiente.
- ☐ Aplicación de skybox, para crear un ambiente similar al de la referencia.
- ☐ Iluminación del ambiente.

## Limitantes.

Algunas limitaciones en la realización de este proyecto fueron las siguientes.

- La falta de conocimientos en nuestra herramienta de modelado, en este caso se utilizó Maya.
- Al momento de subir el proyecto a github, se pasaba el tamaño permitido.
- En github se tuvieron que comprimir algunos .obj, ya que superan los 25MB.
- Al momento de realizar el ejecutable, las animaciones aumentaron su velocidad considerablemente.

Diagrama de Gantt.

Nombre de la tarea	Fecha de inicio	Fecha de finalización	Asignado	Estado	08.07.2021	09.07.2021	10.07.2021	11.07.2021	12.07.2021	13.07.2021	14.07.2021	15.07.2021	16.07.2021	17.07.2021	18.07.2021	19.07.2021	20.07.2021	21.07.2021	22.07.2021	23.07.2021	24.07.2021	25.07.2021	26.07.2021	27.07.2021
PROYECTO Final	08.07.2021	27.07.2020	Erick Guerra	Terminado																				
Creación de fachada	09.07.2021	10.07.2021	Erick Guerra	Terminado																				
Importación de objetos	10.07.2021	11.07.2021	Erick Guerra	Terminado																				
Texturación de objetos y facl	11.07.2021	12.07.2021	Erick Guerra	Terminado																				
Exportación de objetos	12.07.2021	14.07.2021	Erick Guerra	Terminado																				
Acomodo de objetos en open	14.07.2021	16.07.2021	Erick Guerra	Terminado																				
Animación de objetos	17.07.2021	18.07.2021	Erick Guerra	Terminado																				
Animación con keyframes	19.07.2021	22.07.2021	Erick Guerra	Terminado																				
Iluminación	22.07.2021	23.07.2021	Erick Guerra	Terminado																				
SkyBox	23.07.2021	25.07.2021	Erick Guerra	Terminado																				
Manual De Usuario	24.07.2021	26.07.2021	Erick Guerra	Terminado																				
Manual Técnico	26.07.2021	27.07.2021	Erick Guerra	Terminado																				

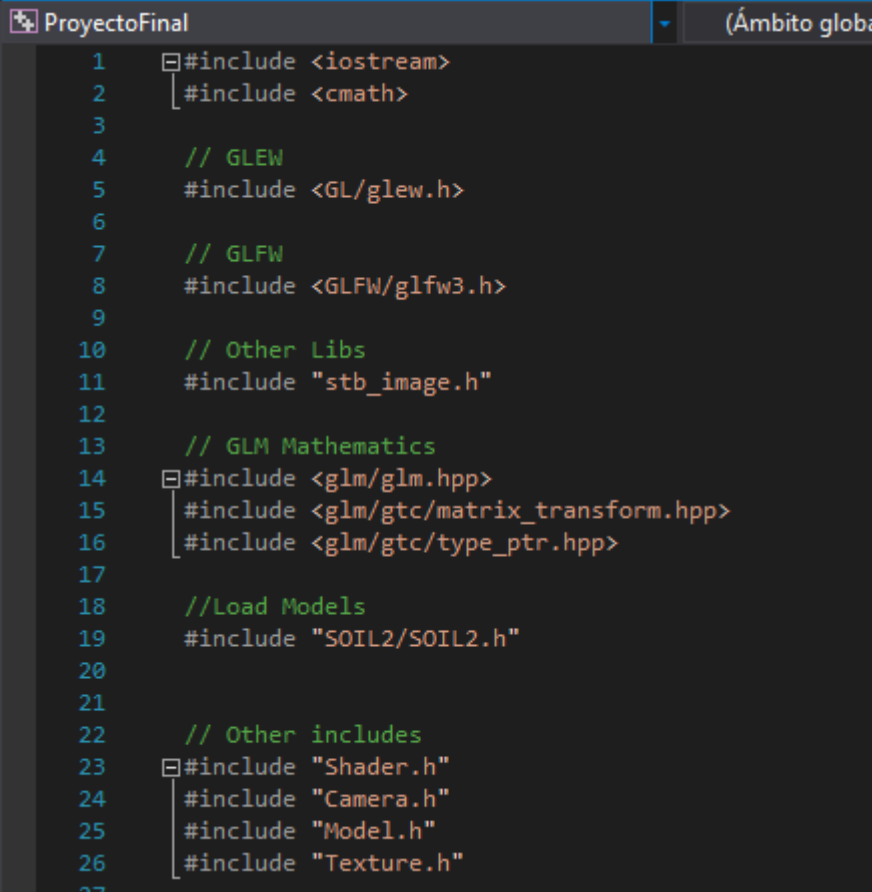
## Documentación del código.

Librerías utilizadas en el código.

- **GLFW**: Capacidad de crear y dirigir ventanas y aplicaciones OpenGL, así como recibir la entrada de joystick, teclado y ratón.
- **GLEW**: Ayuda en la carga y consulta de extensiones de OpenGL.
- **glm**: Cálculos matemáticos que permite aplicar operaciones de transformación en los modelos.
- **assimp**: Es una librería que nos servirá para cargar modelos o escenas 3D almacenados en gran variedad de formatos.

Nuestro archivo principal es el llamado “ProyectoFinal.cpp”, a continuación se describirán algunas de las partes más importantes de este código.

Cargamos nuestras librerías.



```
1  #include <iostream>
2  #include <cmath>
3
4  // GLEW
5  #include <GL/glew.h>
6
7  // GLFW
8  #include <GLFW/glfw3.h>
9
10 // Other Libs
11 #include "stb_image.h"
12
13 // GLM Mathematics
14 #include <glm/glm.hpp>
15 #include <glm/gtc/matrix_transform.hpp>
16 #include <glm/gtc/type_ptr.hpp>
17
18 //Load Models
19 #include "SOIL2/SOIL2.h"
20
21
22 // Other includes
23 #include "Shader.h"
24 #include "Camera.h"
25 #include "Model.h"
26 #include "Texture.h"
27
```

Colocamos la posición inicial en donde se encontrará nuestra cámara.

```

38 // Camera posición
39 Camera camera(glm::vec3(0.0f, 2.0f, 20.0f));
40 GLfloat lastX = WIDTH / 2.0;
41 GLfloat lastY = HEIGHT / 2.0;
42 bool keys[1024];
43 bool firstMouse = true;
44 float range = 0.0f;
45 float rot = 0.0f;

```

Colocamos la posición inicial de nuestro personaje.

```

48 // Light attributes
49 glm::vec3 lightPos(0.0f, 0.0f, 0.0f);
50 glm::vec3 PosIni(0.0f, 0.0f, 0.0f); //Posición de nuestro personaje
51 bool active;

```

Línea 65-80 variables para guardar los keyframes.

```

59 float posX = PosIni.x, posY = PosIni.y, posZ = PosIni.z, rotArma = 0, MovYVar = 0, rotBraIzq = 0, rotBraDer = 0;
60
61 #define MAX_FRAMES_Timmy 9//numero total de frames
62 int i_max_steps = 190;
63 int i_curr_steps = 0;//inicio
64 typedef struct _frame
65 {
66     //Variables para GUARDAR Key Frames
67     float posX; //Variable para PosicionX
68     float posY; //Variable para PosicionY
69     float posZ; //Variable para PosicionZ
70     float incX; //Variable para IncrementoX
71     float incY; //Variable para IncrementoY
72     float incZ; //Variable para IncrementoZ

```

Posición de nuestros point light.

```

88 // Posición de nuestras luces
89 glm::vec3 pointLightPositions[] = {
90     glm::vec3(posX,posY,posZ),
91     glm::vec3(0,9.1f,0),
92     glm::vec3(8.5f,9.1f,0),
93     glm::vec3(-11.53f,6.0f,0)
94 };

```

Variables para animaciones.

```

100  //ANIMACIONES
101  //Animación de la silla
102  float rotSilla = 0.0;
103  float rotPuerta = 0.0;
104  float MovmouseX = 0.55;
105  float MovmouseY = -0.95;
106  float MovmouseZ = -2.5;
107
108  bool circuitosilla = false;
109  bool recorrido1silla = true;
110  bool recorrido2silla = true;
111
112  bool circuitoPuerta = false;
113  bool recorrido1Puerta = true;
114  bool recorrido2Puerta = true;
115
116  bool circuitoMouse = false;
117  bool recorrido1Mouse = true;
118  bool recorrido2Mouse = true;
119  bool recorrido3Mouse = true;

```

Línea 122-156.

Datos guardados para los keyframes.

```

122  void saveFrame(void)
123  {
124
125      KeyFrame[0].rotArma = 0;
126      KeyFrame[1].rotArma = 80;
127      KeyFrame[2].rotArma = -85;
128      KeyFrame[3].rotArma = 0;
129      KeyFrame[4].rotArma = 80;
130      KeyFrame[5].rotArma = -85;
131      KeyFrame[6].rotArma = 0;
132

```

Reseteo de keyframes.

```

158  void resetElements(void)
159  {
160      posX = KeyFrame[0].posX;
161      posY = KeyFrame[0].posY;
162      posZ = KeyFrame[0].posZ;
163
164      rotArma = KeyFrame[0].rotArma; //Reseteo de la arma
165      MovYVar = KeyFrame[0].MovYVar; //Reseteo de varita
166      rotBraIzq = KeyFrame[0].rotBraIzq; //Reseteo brazo izquierdo
167      rotBraDer = KeyFrame[0].rotBraDer; //Reseteo brazo derecho
168
169  }

```

Interpolación, esta función nos ayudará a la interpolación con las posiciones guardadas en nuestra función saveFrame()

```

171 void interpolation(void)
172 {
173
174     KeyFrame[playIndex].incX = (KeyFrame[playIndex + 1].posX - KeyFrame[playIndex].posX) / i_max_steps;
175     KeyFrame[playIndex].incY = (KeyFrame[playIndex + 1].posY - KeyFrame[playIndex].posY) / i_max_steps;
176     KeyFrame[playIndex].incZ = (KeyFrame[playIndex + 1].posZ - KeyFrame[playIndex].posZ) / i_max_steps;
177
178     KeyFrame[playIndex].rotInc = (KeyFrame[playIndex + 1].rotArma - KeyFrame[playIndex].rotArma) / i_max_steps; //Arma
179     KeyFrame[playIndex].rotInc2 = (KeyFrame[playIndex + 1].MovYVar - KeyFrame[playIndex].MovYVar) / i_max_steps; //Varita
180     KeyFrame[playIndex].rotInc3 = (KeyFrame[playIndex + 1].rotBraIzq - KeyFrame[playIndex].rotBraIzq) / i_max_steps; //Brazo izquierdo
181     KeyFrame[playIndex].rotInc4 = (KeyFrame[playIndex + 1].rotBraDer - KeyFrame[playIndex].rotBraDer) / i_max_steps; //Brazo derecho
182 }

```

## Carga de shaders.

```

239 Shader lightingShader("Shaders/lighting.vs", "Shaders/lighting.frag");
240 Shader lampShader("Shaders/lamp.vs", "Shaders/lamp.frag");
241 Shader SkyBoxshader("Shaders/SkyBox.vs", "Shaders/SkyBox.frag");
242

```

## Carga de archivos .obj

```

243 //Partes del personaje
244 Model Torso((char*)"Models/Superheroe/Cuerpo.obj");
245 Model BrazoIzq((char*)"Models/Superheroe/BraIzq.obj");
246 Model BrazoDer((char*)"Models/Superheroe/BraDer.obj");
247 Model Arma((char*)"Models/Superheroe/Arma.obj");
248 Model Varita((char*)"Models/Varita/Varita.obj");
249
250 //Objetos de la casa
251 Model Casa( (char *)"Models/Casa/Casa.obj");
252 Model Silla((char*)"Models/Silla/Silla.obj");
253 Model Escritorio((char*)"Models/Escritorio/Escritorio.obj");
254 Model Cama((char*)"Models/Cama/Cama.obj");
255 Model Buro((char*)"Models/Buro/Buro.obj");
256 Model Cuadro((char*)"Models/Cuadro/Cuadro.obj");
257 Model Pecera((char*)"Models/Pecera/Pecera.obj");
258 Model Puerta((char*)"Models/Puerta/Puerta.obj");
259 Model Computadora((char*)"Models/Computadora/Computadora.obj");
260 Model Mousee((char*)"Models/Mouse/Mouse.obj");
261 Model Techo((char*)"Models/Techo/Techo.obj");
262 Model Muro((char*)"Models/Muro/Muro.obj");
263
264 //Personajes
265 Model Timmy((char*)"Models/Timmy/Timmy.obj");
266

```

## inicialización de keyframes.

```

271 for (int i = 0; i < MAX_FRAMES_Timmy; i++)
272 {
273
274     KeyFrame[i].rotArma = 0;
275     KeyFrame[i].rotInc = 0;
276     KeyFrame[i].MovYVar = 0; //Rodilla Derecha
277     KeyFrame[i].rotInc2 = 0;
278     KeyFrame[i].rotBraIzq = 0; //Brazo Izquierdo
279     KeyFrame[i].rotInc3 = 0;
280     KeyFrame[i].rotBraDer = 0; //Brazo Derecho
281     KeyFrame[i].rotInc4 = 0;
282 }
283

```



Línea 287-331 vértices.

```
287     GLfloat vertices[] =
288     {
289         // Positions           // Normals           // Texture Coords
290         -0.5f, -0.5f, -0.5f,    0.0f,  0.0f, -1.0f,    0.0f,  0.0f,
291         0.5f, -0.5f, -0.5f,    0.0f,  0.0f, -1.0f,    1.0f,  0.0f,
292         0.5f,  0.5f, -0.5f,    0.0f,  0.0f, -1.0f,    1.0f,  1.0f,
293         0.5f,  0.5f, -0.5f,    0.0f,  0.0f, -1.0f,    1.0f,  1.0f,
294         -0.5f,  0.5f, -0.5f,    0.0f,  0.0f, -1.0f,    0.0f,  1.0f,
295         -0.5f, -0.5f, -0.5f,    0.0f,  0.0f, -1.0f,    0.0f,  0.0f,
```

Línea 334-377 SkyBox.

```
334     GLfloat skyboxVertices[] = {
335         // Positions
336         -1.0f,  1.0f, -1.0f,
337         -1.0f, -1.0f, -1.0f,
338         1.0f, -1.0f, -1.0f,
339         1.0f, -1.0f, -1.0f,
340         1.0f,  1.0f, -1.0f,
341         -1.0f,  1.0f, -1.0f,
```

Posición de nuestro contenedor.

```
394     glm::vec3 cubePositions[] = {
395         glm::vec3(0.0f,  0.0f,  0.0f),
396         glm::vec3(2.0f,  5.0f, -15.0f),
397         glm::vec3(-1.5f, -2.2f, -2.5f),
398         glm::vec3(-3.8f, -2.0f, -12.3f),
399         glm::vec3(2.4f, -0.4f, -3.5f),
400         glm::vec3(-1.7f,  3.0f, -7.5f),
401         glm::vec3(1.3f, -2.0f, -2.5f),
402         glm::vec3(1.5f,  2.0f, -2.5f),
403         glm::vec3(1.5f,  0.2f, -1.5f),
404         glm::vec3(-1.3f,  1.0f, -1.5f)
405     };
```

Se reserva memoria para el arreglo de vértices (VAO) y los buffers (VBO y EBO).

```
409     GLuint VBO, VAO, EBO;
410     glGenVertexArrays(1, &VAO);
411     glGenBuffers(1, &VBO);
412     glGenBuffers(1, &EBO);
413
414     glBindVertexArray(VAO);
415     glBindBuffer(GL_ARRAY_BUFFER, VBO);
416     glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);
417
418     glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO);
419     glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices), indices, GL_STATIC_DRAW);
420
```

Se indican las posiciones, las normales y las coordenadas de texturas.

```

421 // Position attribute
422 glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 * sizeof(GLfloat), (GLvoid *)0);
423 glEnableVertexAttribArray(0);
424 // Normals attribute
425 glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 8 * sizeof(GLfloat), (GLvoid *) (3 * sizeof(GLfloat)));
426 glEnableVertexAttribArray(1);
427 // Texture Coordinate attribute
428 glVertexAttribPointer(2, 2, GL_FLOAT, GL_FALSE, 8 * sizeof(GLfloat), (GLvoid *) (6 * sizeof(GLfloat)));
429 glEnableVertexAttribArray(2);
430 glBindVertexArray(0);

```

Obtenemos los vértices que utilizará nuestra SkyBox.

```

444 //SkyBox
445 GLuint skyboxVBO, skyboxVAO;
446 glGenVertexArrays(1, &skyboxVAO);
447 glGenBuffers(1, &skyboxVBO);
448 glBindVertexArray(skyboxVAO);
449 glBindBuffer(GL_ARRAY_BUFFER, skyboxVBO);
450 glBufferData(GL_ARRAY_BUFFER, sizeof(skyboxVertices), &skyboxVertices, GL_STATIC_DRAW);
451 glEnableVertexAttribArray(0);
452 glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(GLfloat), (GLvoid *)0); //Obtener vertices para trbajar con ellos
453

```

Cargamos las textura de nuestro skyBox.

```

455 vector<const GLchar*> faces; //Arreglo de caras
456 faces.push_back("SkyBox/negz.tga");
457 faces.push_back("SkyBox/negx.tga");
458 faces.push_back("SkyBox/posy.tga");
459 faces.push_back("SkyBox/negy.tga");
460 faces.push_back("SkyBox/posx.tga");
461 faces.push_back("SkyBox/posz.tga");
462

```

Indicamos el tipo de proyección de nuestro proyecto.

```

464
465 glm::mat4 projection = glm::perspective(camera.GetZoom(), (GLfloat)SCREEN_WIDTH / (GLfloat)SCREEN_HEIGHT, 0.1f, 1000.0f);
466

```

Línea 500-554, acomodo de nuestras spot light.

```

500 glUniform3f(glGetUniformLocation(lightningShader.Program, "dirLight.direction"), -0.2f, -1.0f, -0.3f);
501 glUniform3f(glGetUniformLocation(lightningShader.Program, "dirLight.ambient"), 0.3f, 0.3f, 0.3f);
502 glUniform3f(glGetUniformLocation(lightningShader.Program, "dirLight.diffuse"), 0.4f, 0.4f, 0.4f);
503 glUniform3f(glGetUniformLocation(lightningShader.Program, "dirLight.specular"), 0.5f, 0.5f, 0.5f);
504
505 // Point light 1
506
507 glUniform3f(glGetUniformLocation(lightningShader.Program, "pointLights[0].position"), pointLightPositions[0].x, pointLightPosi
508 glUniform3f(glGetUniformLocation(lightningShader.Program, "pointLights[0].ambient"), 0.05f, 0.05f, 0.05f);
509 glUniform3f(glGetUniformLocation(lightningShader.Program, "pointLights[0].diffuse"), LightP1.x, LightP1.y, LightP1.z);
510 glUniform3f(glGetUniformLocation(lightningShader.Program, "pointLights[0].specular"), LightP1.x, LightP1.y, LightP1.z);
511 glUniform1f(glGetUniformLocation(lightningShader.Program, "pointLights[0].constant"), 1.0f);
512 glUniform1f(glGetUniformLocation(lightningShader.Program, "pointLights[0].linear"), 0.09f);
513 glUniform1f(glGetUniformLocation(lightningShader.Program, "pointLights[0].quadratic"), 0.032f);
514

```

Declaramos nuestras matrices de vista, proyección y modelo.

```

564 // Get the uniform locations
565 GLint modelLoc = glGetUniformLocation(lightningShader.Program, "model");
566 GLint viewLoc = glGetUniformLocation(lightningShader.Program, "view");
567 GLint projLoc = glGetUniformLocation(lightningShader.Program, "projection");
568

```

Línea 586-749, carga de modelos, con su respectiva traslación, rotación y reseteo, según el caso.

```

586 //CARGA DE MODELOS
587 //Personaje*****
588 view = camera.GetViewMatrix();
589 glm::mat4 model(1);
590 tmp = model = glm::translate(model, glm::vec3(0, 1, 0));
591 model = glm::translate(model, glm::vec3(posX, posY, posZ));
592 model = glm::translate(model, glm::vec3(0.0f, -1.55f, -0.9f));
593 //model = glm::rotate(model, glm::radians(rot), glm::vec3(0.0f, 1.0f, 0.0f));
594 glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
595 Torso.Draw(lightningShader);
596
597 //Brazo Izquierdo
598 view = camera.GetViewMatrix();
599 model = glm::mat4(1);
600 model = glm::translate(model, glm::vec3(posX, posY, posZ));
601 //model = glm::rotate(model, glm::radians(rot), glm::vec3(0.0f, 1.0f, 0.0f));
602 model = glm::translate(model, glm::vec3(0.4f, -0.5f, -0.78f));
603 model = glm::rotate(model, glm::radians(-rotBraIzq), glm::vec3(0.0f, 1.0f, 0.0f)); //Asignar moviento
604 glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
605 BrazoIzq.Draw(lightningShader);

```

**Dibujamos nuestras spot light.**

```

765 // Draw the light object (using light's vertex attributes)
766 glBindVertexArray(lightVAO);
767 for (GLuint i = 0; i < 4; i++)
768 {
769     model = glm::mat4(1);
770     model = glm::translate(model, pointLightPositions[i]);
771     model = glm::scale(model, glm::vec3(0.2f)); // Make it a smaller cube
772     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
773     glDrawArrays(GL_TRIANGLES, 0, 36);
774 }
775 glBindVertexArray(0);

```

**Dibujamos nuestra Sky box e iniciamos nuestra función saveFrame.**

```

785 // skybox cube
786 glBindVertexArray(skyboxVAO);
787 glActiveTexture(GL_TEXTURE1);
788 glBindTexture(GL_TEXTURE_CUBE_MAP, cubemapTexture);
789 glDrawArrays(GL_TRIANGLES, 0, 36);
790 glBindVertexArray(0);
791 glDepthFunc(GL_LESS); // Set depth function back to default
792
793
794
795
796 // Swap the screen buffers
797 glfwSwapBuffers(window);
798
799 //Iniciamos nuestra fucncion con los keyframes ya cargados
800 saveFrame();

```

**Línea 822-895, animaciones de los diferentes objetos.**

```

821 //Animaciones
822 void animacion()
823 {
824     //Movimiento de la silla
825     if (circuitosilla) {
826         if (recorrido1silla) {
827             rotSilla += 1.0f;
828             if (rotSilla == 180.0f) {
829                 recorrido1silla = false;
830                 recorrido2silla = true;
831             }
832         }
833
834         if (recorrido2silla) {
835             rotSilla -= 1.0f;
836             if (rotSilla == 0.0f) {
837                 recorrido1silla = true;
838                 recorrido2silla = false;
839             }
840         }
841     }
842 }
843

```

Animación de nuestro personaje, a partir de los keyframes.

```

898 if (playT)
899 {
900     if (i_curr_steps >= i_max_steps) //end of animation between frames
901     {
902         playIndex++;
903         if (playIndex > FrameIndex - 2) //end of total animation?
904         {
905             printf("termina anim\n");
906             playIndex = 0;
907             playT = false;
908         }
909         else //Next frame interpolations
910         {
911             i_curr_steps = 0; //Reset counter
912             //Interpolation
913             interpolation();
914         }
915     }
916     else
917     {
918
919
920         rotArma += KeyFrame[playIndex].rotInc; //Rotacion del arma
921         MovYVar += KeyFrame[playIndex].rotInc2; //Rodilla derecha

```

Indicamos que la tecla “L”, será la encargada de iniciar nuestras animaciones con keyframes.

```

933 void KeyCallback(GLFWwindow *window, int key, int scancode, int action, int mode)
934 {
935     if (keys[GLFW_KEY_L])
936     {
937         if (playT == false && (FrameIndex > 1))
938         {
939             resetElements();
940             //First Interpolation
941             interpolation();
942             playT = true;
943             playIndex = 0;
944             i_curr_steps = 0;
945         }
946         else
947         {
948             playT = false;
949         }
950     }
951 }
952
953

```

**Cerrar nuestro proyecto.**

```

965     if (GLFW_KEY_ESCAPE == key && GLFW_PRESS == action)
966     {
967         glfwSetWindowShouldClose(window, GL_TRUE);
968     }

```

Indicamos que nuestra barra espaciadora encenderá nuestra luz que se encuentra en el centro con tonalidad roja.

```

982     if (keys[GLFW_KEY_SPACE])
983     {
984         active = !active;
985         if (active)
986             LightP1 = glm::vec3(1.0f, 0.0f, 0.0f);
987         else
988             LightP1 = glm::vec3(0.0f, 0.0f, 0.0f);
989     }
990 }

```

Indicamos que al momento de abrir el proyecto, nuestro mouse será el encargado de mover la cámara en las diferentes posiciones Y, X y Z, complementandose con las teclas que mueven la cámara.

```

992 void MouseCallback(GLFWwindow *window, double xPos, double yPos)
993 {
994     if (firstMouse)
995     {
996         lastX = xPos;
997         lastY = yPos;
998         firstMouse = false;
999     }
1000
1001     GLfloat xOffset = xPos - lastX;
1002     GLfloat yOffset = lastY - yPos; // Reversed since y-coordinates go from bottom to left
1003
1004     lastX = xPos;
1005     lastY = yPos;
1006
1007     camera.ProcessMouseMovement(xOffset, yOffset);
1008 }
1009

```

Línea 1012-1129, asignamos las teclas que iniciaran o detendrán el movimiento de nuestras animaciones, que no están hechas con keyframes.

```

1012 void DoMovement()
1013 {
1014     //Activar el movimiento de la Silla
1015     if (keys[GLFW_KEY_Z]) {
1016         circuitosilla = true;
1017     }
1018     if (keys[GLFW_KEY_X]) {
1019         circuitosilla = false;
1020     }
1021
1022     //Activar el movimiento de la Puerta
1023     if (keys[GLFW_KEY_C]) {
1024         circuitoPuerta = true;
1025     }
1026     if (keys[GLFW_KEY_V]) {
1027         circuitoPuerta = false;
1028     }
1029
1030
1031     //Activar el movimiento del Mouse
1032     if (keys[GLFW_KEY_B]) {
1033         circuitoMouse = true;
1034     }
1035

```

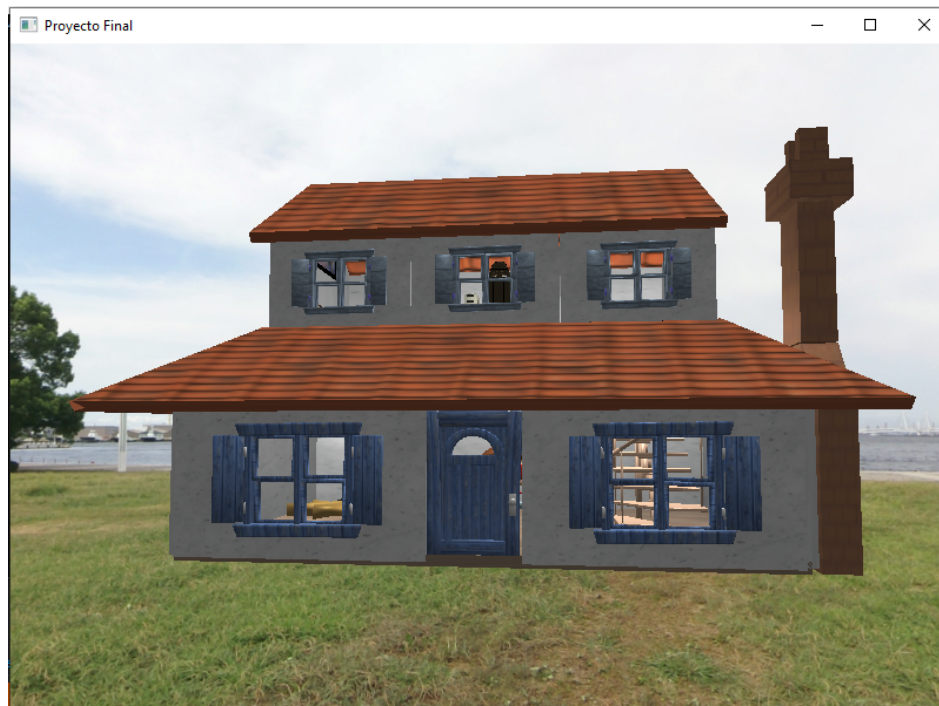
Asignamos las teclas que moverán a nuestra cámara.

```

1134 // Camera controls
1135 if (keys[GLFW_KEY_W] || keys[GLFW_KEY_UP])
1136 {
1137     camera.ProcessKeyboard(FORWARD, deltaTime);
1138 }
1139
1140
1141 if (keys[GLFW_KEY_S] || keys[GLFW_KEY_DOWN])
1142 {
1143     camera.ProcessKeyboard(BACKWARD, deltaTime);
1144 }
1145
1146
1147
1148 if (keys[GLFW_KEY_A] || keys[GLFW_KEY_LEFT])
1149 {
1150     camera.ProcessKeyboard(LEFT, deltaTime);
1151 }
1152
1153
1154
1155 if (keys[GLFW_KEY_D] || keys[GLFW_KEY_RIGHT])
1156 {
1157     camera.ProcessKeyboard(RIGHT, deltaTime);
1158 }

```

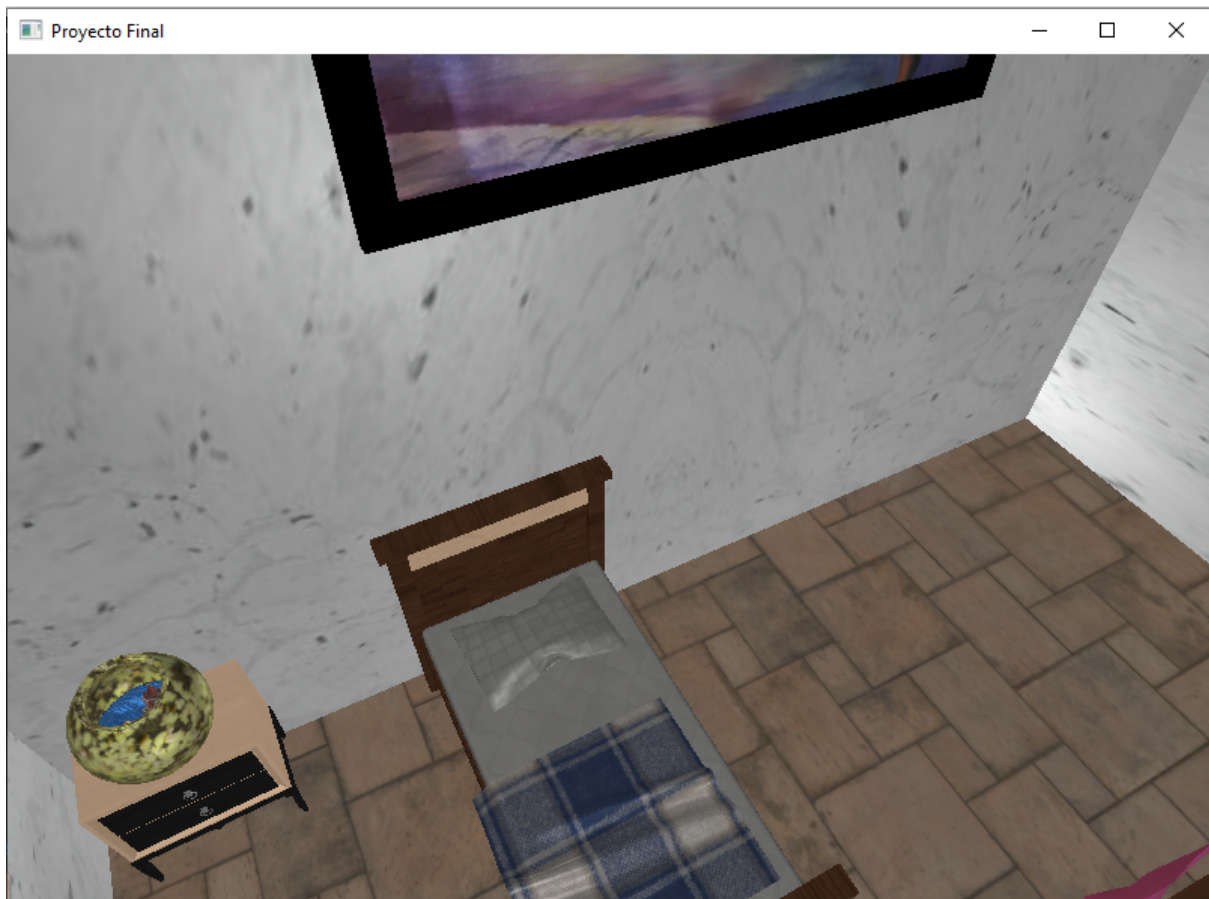
## Funcionamiento del proyecto.











## Conclusión.

Al término de este proyecto pudimos completar los objetivos cumplidos al inicio del mismo, además queda demostrado el uso de las aplicaciones de las diferentes herramientas mostradas en el laboratorio de Computación Gráfica e Interacción Humano-Computadora.