

```
1  /**
2   * @file 419048901_Proyecto_GP004.cpp
3   * @brief Archivo principal CPP (main program) del proyecto
4   * @author NumCuenta: 419048901
5   * @date 11/05/2022
6   */
7
8  // Operaciones E/S
9  #include <iostream>
10
11 // Operaciones Matematicas
12 #include <cmath>
13
14 // GLEW
15 #include <GL/glew.h>
16
17 // GLFW
18 #include <GLFW/glfw3.h>
19
20 // Other Libs
21 #include "stb_image.h"
22
23 // GLM Mathematics
24 #include <glm/glm.hpp>
25 #include <glm/gtc/matrix_transform.hpp>
26 #include <glm/gtc/type_ptr.hpp>
27
28 //Load Models
29 #include "SOIL2/SOIL2.h"
30
31 // Other includes
32 #include "Shader.h"
33 #include "Camera.h"
34 #include "Model.h"
35 #include "Texture.h"
36 #include "modelAnim.h"
37
38 // Function prototypes
39 void KeyCallback(GLFWwindow *window, int key, int scancode, int action, int mode);
40 void MouseCallback(GLFWwindow *window, double xPos, double yPos);
41 void DoMovement();
42 void animacion();
43
44 // Window dimensions
45 const GLuint WIDTH = 800, HEIGHT = 600;
46 int SCREEN_WIDTH, SCREEN_HEIGHT;
47
48 // Camera
```

```
49 Camera camera(glm::vec3(0.0f, 10.0f, 25.0f));
50 GLfloat lastX = WIDTH / 2.0;
51 GLfloat lastY = HEIGHT / 2.0;
52 bool keys[1024];
53 bool firstMouse = true;
54
55
56 // Light attributes
57 glm::vec3 lightPos(0.0f, 0.0f, 0.0f);
58 glm::vec3 PosIni(-16.0f, 1.0f, -70.0f);
59 glm::vec3 lightDirection(0.0f, -1.0f, -1.0f);
60 bool active;
61
62 bool encendido = false;
63 // Positions of the point lights
64 glm::vec3 pointLightPositions[] = {
65     glm::vec3(0.0f, 19.0f, 0.0f)
66 };
67
68 // Position of the SpotLight
69 glm::vec3 spotLightPosition = glm::vec3(0.0f, 19.0f, 0.0f);
70
71 int dir = 0;
72 // Directions of the SpotLight
73 glm::vec3 spotLightDir[] = {
74     glm::vec3(0.0f, -1.0f, 0.0f), // Abajo
75     glm::vec3(1.0f, 0.0f, 0.0f), // Derecha
76     glm::vec3(0.0f, 0.0f, -1.0f), // Atras
77     glm::vec3(-1.0f, 0.0f, 0.0f), // Izquierda
78     glm::vec3(0.0f, 0.0f, 1.0f), // Frente
79     glm::vec3(0.0f, 1.0f, 0.0f), // Arriba
80     glm::vec3(0.0f, -1.0f, 0.0f) // Abajo
81 };
82
83 float vertices[] = {
84     -0.5f, -0.5f, -0.5f, 0.0f, 0.0f, -1.0f,
85     0.5f, -0.5f, -0.5f, 0.0f, 0.0f, -1.0f,
86     0.5f, 0.5f, -0.5f, 0.0f, 0.0f, -1.0f,
87     0.5f, 0.5f, -0.5f, 0.0f, 0.0f, -1.0f,
88     -0.5f, 0.5f, -0.5f, 0.0f, 0.0f, -1.0f,
89     -0.5f, -0.5f, -0.5f, 0.0f, 0.0f, -1.0f,
90
91     -0.5f, -0.5f, 0.5f, 0.0f, 0.0f, 1.0f,
92     0.5f, -0.5f, 0.5f, 0.0f, 0.0f, 1.0f,
93     0.5f, 0.5f, 0.5f, 0.0f, 0.0f, 1.0f,
94     0.5f, 0.5f, 0.5f, 0.0f, 0.0f, 1.0f,
95     -0.5f, 0.5f, 0.5f, 0.0f, 0.0f, 1.0f,
96     -0.5f, -0.5f, 0.5f, 0.0f, 0.0f, 1.0f,
97 }
```

```
98      -0.5f,  0.5f,  0.5f, -1.0f,  0.0f,  0.0f,
99      -0.5f,  0.5f, -0.5f, -1.0f,  0.0f,  0.0f,
100     -0.5f, -0.5f, -0.5f, -1.0f,  0.0f,  0.0f,
101     -0.5f, -0.5f, -0.5f, -1.0f,  0.0f,  0.0f,
102     -0.5f, -0.5f,  0.5f, -1.0f,  0.0f,  0.0f,
103     -0.5f,  0.5f,  0.5f, -1.0f,  0.0f,  0.0f,
104
105         0.5f,  0.5f,  0.5f,  1.0f,  0.0f,  0.0f,
106         0.5f,  0.5f, -0.5f,  1.0f,  0.0f,  0.0f,
107         0.5f, -0.5f, -0.5f,  1.0f,  0.0f,  0.0f,
108         0.5f, -0.5f, -0.5f,  1.0f,  0.0f,  0.0f,
109         0.5f, -0.5f,  0.5f,  1.0f,  0.0f,  0.0f,
110         0.5f,  0.5f,  0.5f,  1.0f,  0.0f,  0.0f,
111
112     -0.5f, -0.5f, -0.5f,  0.0f, -1.0f,  0.0f,
113         0.5f, -0.5f, -0.5f,  0.0f, -1.0f,  0.0f,
114         0.5f, -0.5f,  0.5f,  0.0f, -1.0f,  0.0f,
115         0.5f, -0.5f,  0.5f,  0.0f, -1.0f,  0.0f,
116     -0.5f, -0.5f,  0.5f,  0.0f, -1.0f,  0.0f,
117     -0.5f, -0.5f, -0.5f,  0.0f, -1.0f,  0.0f,
118
119     -0.5f,  0.5f, -0.5f,  0.0f,  1.0f,  0.0f,
120         0.5f,  0.5f, -0.5f,  0.0f,  1.0f,  0.0f,
121         0.5f,  0.5f,  0.5f,  0.0f,  1.0f,  0.0f,
122         0.5f,  0.5f,  0.5f,  0.0f,  1.0f,  0.0f,
123     -0.5f,  0.5f,  0.5f,  0.0f,  1.0f,  0.0f,
124     -0.5f,  0.5f, -0.5f,  0.0f,  1.0f,  0.0f
125 };
126
127 GLfloat skyboxVertices[] = {
128     // Positions
129     -1.0f,  1.0f, -1.0f,
130     -1.0f, -1.0f, -1.0f,
131     1.0f, -1.0f, -1.0f,
132     1.0f, -1.0f, -1.0f,
133     1.0f,  1.0f, -1.0f,
134     -1.0f,  1.0f, -1.0f,
135
136     -1.0f, -1.0f,  1.0f,
137     -1.0f, -1.0f, -1.0f,
138     -1.0f,  1.0f, -1.0f,
139     -1.0f,  1.0f, -1.0f,
140     -1.0f,  1.0f,  1.0f,
141     -1.0f, -1.0f,  1.0f,
142
143     1.0f, -1.0f, -1.0f,
144     1.0f, -1.0f,  1.0f,
145     1.0f,  1.0f,  1.0f,
146     1.0f,  1.0f,  1.0f,
```

```
147     1.0f,  1.0f, -1.0f,
148     1.0f, -1.0f, -1.0f,
149
150     -1.0f, -1.0f,  1.0f,
151     -1.0f,  1.0f,  1.0f,
152     1.0f,  1.0f,  1.0f,
153     1.0f,  1.0f,  1.0f,
154     1.0f, -1.0f,  1.0f,
155     -1.0f, -1.0f,  1.0f,
156
157     -1.0f,  1.0f, -1.0f,
158     1.0f,  1.0f, -1.0f,
159     1.0f,  1.0f,  1.0f,
160     1.0f,  1.0f,  1.0f,
161     -1.0f,  1.0f,  1.0f,
162     -1.0f,  1.0f, -1.0f,
163
164     -1.0f, -1.0f, -1.0f,
165     -1.0f, -1.0f,  1.0f,
166     1.0f, -1.0f, -1.0f,
167     1.0f, -1.0f, -1.0f,
168     -1.0f, -1.0f,  1.0f,
169     1.0f, -1.0f,  1.0f
170 };
171
172 glm::vec3 Light1 = glm::vec3(0);
173 glm::vec3 Light2 = glm::vec3(0);
174 glm::vec3 Light3 = glm::vec3(0);
175 glm::vec3 Light4 = glm::vec3(0);
176
177 /**
178  * \var rotDor, actionDoor, openDoor
179  * \brief Variables Animación de Puerta
180  */
181 float rotDoor = 0.0f;
182 bool actionDoor = false, openDoor = false;
183
184 /**
185  * \var rotCam, CamDerecha
186  * \brief Variables Animación de Camara Seguridad
187  */
188 float rotCam = 0.0;
189 bool CamDerecha = false;
190
191 /**
192  * \var posIniCar, movKitXY, rotKit, circuito, recorridos1-8
193  * \brief Variables Animación del coche
194  */
195 glm::vec3 PosIniCar(80.0f, 0.0f, 14.0f);
```

```
196 float movKitX = 0.0;
197 float movKitZ = 0.0;
198 float rotKit = 0.0;
199
200 bool circuito = false;
201 bool recorrido1 = true; bool recorrido2 = false; bool recorrido3 = false; ➤
    bool recorrido4 = false;
202 bool recorrido5 = false; bool recorrido6 = false; bool recorrido7 = false; ➤
    bool recorrido8 = false;
203
204 /**
205  * \var posIniPerson
206  * \brief Variable Animación del Personaje
207  */
208 glm::vec3 PosIniPerson(-16.0f, 0.0f, -70.0f);
209
210 // Deltatime
211 GLfloat deltaTime = 0.0f; // Time between current frame and last frame
212 GLfloat lastFrame = 0.0f; // Time of last frame
213
214
215 /**
216  * \fn int main()
217  * \brief Funcion del programa principal
218  * \return Devuelve 0 de programa exitoso
219  */
220 int main()
221 {
222     // Init GLFW
223     glfwInit();
224
225     // Set all the required options for GLFW
226     /*glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
227     glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
228     glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
229     glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
230     glfwWindowHint(GLFW_RESIZABLE, GL_FALSE);*/
231
232     // Create a GLFWwindow object that we can use for GLFW's functions
233     GLFWwindow* window = glfwCreateWindow(WIDTH, HEIGHT, "Proyecto ➤
        Gimnasio \"FORM\" : 419048901 - GP004", nullptr, nullptr);
234
235     if (nullptr == window)
236     {
237         std::cout << "Failed to create GLFW window" << std::endl;
238         glfwTerminate();
239
240         return EXIT_FAILURE;
241     }
```

```
242
243     glfwMakeContextCurrent(window);
244     glfwGetFramebufferSize(window, &SCREEN_WIDTH, &SCREEN_HEIGHT);
245
246     // Set the required callback functions
247     glfwSetKeyCallback(window, KeyCallback);
248     glfwSetCursorPosCallback(window, MouseCallback);
249
250     // GLFW Options
251     //glfwSetInputMode(window, GLFW_CURSOR, GLFW_CURSOR_DISABLED);
252
253     // Set this to true so GLEW knows to use a modern approach to      ➤
254     // retrieving function pointers and extensions
255     glewExperimental = GL_TRUE;
256
257     // Initialize GLEW to setup the OpenGL Function pointers
258     if (GLEW_OK != glewInit())
259     {
260         std::cout << "Failed to initialize GLEW" << std::endl;
261         return EXIT_FAILURE;
262     }
263
264     // Define the viewport dimensions
265     glViewport(0, 0, SCREEN_WIDTH, SCREEN_HEIGHT);
266
267     // Carga de Shaders
268     Shader lightingShader("Shaders/lighting.vs", "Shaders/lighting.frag");
269     Shader lampShader("Shaders/lamp.vs", "Shaders/lamp.frag");
270     Shader SkyBoxshader("Shaders/SkyBox.vs", "Shaders/SkyBox.frag");
271     Shader animShader("Shaders/anim.vs", "Shaders/anim.frag");
272
273     // Carga de modelos de gimnasio
274     Model Piso((char*)"Models/Gym/calles.obj");
275     Model Habitacion((char*)"Models/Gym/habitacion.obj");
276     Model Entrada((char*)"Models/Gym/entrada.obj");
277     Model Puerta((char*)"Models/Gym/puertaPrincipal.obj");
278     Model Estante((char*)"Models/Gym/estante.obj");
279     Model Ventanas((char*)"Models/Gym/ventanas.obj");
280     Model Marcos((char*)"Models/Gym/marcos.obj");
281     Model Hidrante((char*)"Models/Ambiente/hidrante.obj");
282
283     Model Banca_inclinada((char*)"Models/Gym/banca_inclinada.obj");
284     Model Caminadora((char*)"Models/Gym/caminadora.obj");
285     Model Barra((char*)"Models/Gym/barra.obj");
286     Model Rack((char*)"Models/Gym/rack.obj");
287     Model Rack2((char*)"Models/Gym/rack2.obj");
288     Model Multi((char*)"Models/Gym/barraMulti.obj");
289
290     Model Mancuerna((char*)"Models/Gym/mancuerna.obj");
```

```
290     Model Mancuerna2((char*)"Models/Gym/mancuerna2.obj");
291     Model Mancuerna3((char*)"Models/Gym/mancuerna3.obj");
292     Model Rusa1((char*)"Models/Gym/rusa1.obj");
293     Model Rusa2((char*)"Models/Gym/rusa2.obj");
294     Model Rusa3((char*)"Models/Gym/rusa3.obj");
295     Model Suiza((char*)"Models/Gym/suiza.obj");
296     Model Gorra1((char*)"Models/Gym/gorra1.obj");
297     Model Gorra2((char*)"Models/Gym/gorra2.obj");
298     Model Gorra3((char*)"Models/Gym/gorra3.obj");
299
300     // Carga de modelos de animación
301     ModelAnim animacionPersonaje("Animaciones/abdominal.dae");
302     Model Carro((char*)"Models/Ambiente/lamborghini.obj");
303     Model Soporte((char*)"Models/Ambiente/soporte.obj");
304     Model Camara((char*)"Models/Ambiente/camara.obj");
305
306
307     // First, set the container's VAO (and VBO)
308     GLuint VBO, VAO;
309     glGenVertexArrays(1, &VAO);
310     glGenBuffers(1, &VBO);
311     glBindVertexArray(VAO);
312     glBindBuffer(GL_ARRAY_BUFFER, VBO);
313     glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,           ➤
314                 GL_STATIC_DRAW);
315     // Position attribute
316     glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(GLfloat), ➤
317                           (GLvoid*)0);
318     glEnableVertexAttribArray(0);
319     // normal attribute
320     glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), ➤
321                           (void*)(3 * sizeof(float)));
322     glEnableVertexAttribArray(1);
323
324     // Set texture units
325     lightingShader.Use();
326     glUniform1i(glGetUniformLocation(lightingShader.Program,           ➤
327                                     "material.diffuse"), 0);
328     glUniform1i(glGetUniformLocation(lightingShader.Program,           ➤
329                                     "material.specular"), 1);
330
331     // SkyBox attributes
332     GLuint skyboxVBO, skyboxVAO;
333     glGenVertexArrays(1, &skyboxVAO);
334     glGenBuffers(1, &skyboxVBO);
335     glBindVertexArray(skyboxVAO);
336     glBindBuffer(GL_ARRAY_BUFFER, skyboxVBO);
337     glBufferData(GL_ARRAY_BUFFER, sizeof(skyboxVertices), &skyboxVertices, ➤
338                 GL_STATIC_DRAW);
```

```
333     glEnableVertexAttribArray(0);
334     glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(GLfloat),
        (GLvoid*)0);
335
336     // Load textures
337     vector<const GLchar*> faces;
338     faces.push_back("SkyBox/right.tga");
339     faces.push_back("SkyBox/left.tga");
340     faces.push_back("SkyBox/top.tga");
341     faces.push_back("SkyBox/bottom.tga");
342     faces.push_back("SkyBox/back.tga");
343     faces.push_back("SkyBox/front.tga");
344
345     GLuint cubemapTexture = TextureLoading::LoadCubemap(faces);
346
347     // Load matrix Projection
348     glm::mat4 projection = glm::perspective(camera.GetZoom(), (GLfloat)
        SCREEN_WIDTH / (GLfloat)SCREEN_HEIGHT, 0.1f, 100.0f);
349
350
351     // Game loop
352     while (!glfwWindowShouldClose(window))
353     {
354
355         // Calculate deltatime of current frame
356         GLfloat currentFrame = glfwGetTime();
357         deltaTime = currentFrame - lastFrame;
358         lastFrame = currentFrame;
359
360         // Check if any events have been activiated (key pressed, mouse
            moved etc.) and call corresponding response functions
361         glfwPollEvents();
362         DoMovement();
363         animacion();
364
365         // Clear the colorbuffer
366         glClearColor(0.1f, 0.1f, 0.1f, 1.0f);
367         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
368
369         // OpenGL options
370         glEnable(GL_DEPTH_TEST);
371
372
373         // Use cooresponding shader when setting uniforms/drawing objects
374         /* ----- Lighting Shader -----*/
375         lightingShader.Use();
376         GLint viewPosLoc = glGetUniformLocation(lightingShader.Program,
            "viewPos");
377         glUniform3f(viewPosLoc, camera.GetPosition().x, camera.GetPosition
```



```
( ).y, camera.GetPosition().z);

378
379 // Directional light
380 glUniform3f(glGetUniformLocation(lightningShader.Program,      ↗
    "dirLight.direction"), 0.2f, -1.0f, -0.3f);
381 glUniform3f(glGetUniformLocation(lightningShader.Program,      ↗
    "dirLight.ambient"), 0.45f, 0.45f, 0.45f); // Luz ambiente + ↗
    DiffuseModify
382 glUniform3f(glGetUniformLocation(lightningShader.Program,      ↗
    "dirLight.diffuse"), 0.1f, 0.1f, 0.1f);
383 glUniform3f(glGetUniformLocation(lightningShader.Program,      ↗
    "dirLight.specular"), 0.35f, 0.35f, 0.35f);

384
385 // Point light
386 glUniform3f(glGetUniformLocation(lightningShader.Program,      ↗
    "pointLights[0].position"), pointLightPositions[0].x,      ↗
    pointLightPositions[0].y, pointLightPositions[0].z);
387 glUniform3f(glGetUniformLocation(lightningShader.Program,      ↗
    "pointLights[0].ambient"), 1.0f, 1.0f, 1.0f);
388 glUniform3f(glGetUniformLocation(lightningShader.Program,      ↗
    "pointLights[0].diffuse"), 1.0f, 1.0f, 1.0f);
389 glUniform3f(glGetUniformLocation(lightningShader.Program,      ↗
    "pointLights[0].specular"), 1.0f, 1.0f, 1.0f);
390 glUniform1f(glGetUniformLocation(lightningShader.Program,      ↗
    "pointLights[0].constant"), 1.0f);
391 glUniform1f(glGetUniformLocation(lightningShader.Program,      ↗
    "pointLights[0].linear"), 0.045f);
392 glUniform1f(glGetUniformLocation(lightningShader.Program,      ↗
    "pointLights[0].quadratic"), 0.0075f);

393
394 // SpotLight GIANT
395 glUniform3f(glGetUniformLocation(lightningShader.Program,      ↗
    "spotLight.position"), spotLightPosition.x, spotLightPosition.y, ↗
    spotLightPosition.z);
396 glUniform3f(glGetUniformLocation(lightningShader.Program,      ↗
    "spotLight.direction"), spotLightDir[dir].x, spotLightDir ↗
    [dir].y, spotLightDir[dir].z);
397 glUniform3f(glGetUniformLocation(lightningShader.Program,      ↗
    "spotLight.ambient"), 0.05f, 0.05f, 0.05f);
398 glUniform3f(glGetUniformLocation(lightningShader.Program,      ↗
    "spotLight.diffuse"), 0.2f, 0.2f, 0.2f);
399 glUniform3f(glGetUniformLocation(lightningShader.Program,      ↗
    "spotLight.specular"), 0.05f, 0.05f, 0.05f);
400 glUniform1f(glGetUniformLocation(lightningShader.Program,      ↗
    "spotLight.constant"), 1.0f);
401 glUniform1f(glGetUniformLocation(lightningShader.Program,      ↗
    "spotLight.linear"), 0.045f);
402 glUniform1f(glGetUniformLocation(lightningShader.Program,      ↗
    "spotLight.quadratic"), 0.0075f);
```

```

...yectoFinal\ProyectoFinal\419048901_Proyecto_Gpo04.cpp 10
403     glUniform1f(glGetUniformLocation(lightningShader.Program, ➤
        "spotLight.cutOff"), glm::cos(glm::radians(12.5f)));
404     glUniform1f(glGetUniformLocation(lightningShader.Program, ➤
        "spotLight.outerCutOff"), glm::cos(glm::radians(15.0f)));
405
406     // Set material properties
407     glUniform1f(glGetUniformLocation(lightningShader.Program, ➤
        "material.shininess"), 32.0f);
408
409     // Create camera transformations
410     glm::mat4 view = camera.GetViewMatrix();
411
412     // Get the uniform locations
413     GLint modelLoc = glGetUniformLocation(lightningShader.Program, ➤
        "model");
414     GLint viewLoc = glGetUniformLocation(lightningShader.Program, ➤
        "view");
415     GLint projLoc = glGetUniformLocation(lightningShader.Program, ➤
        "projection");
416
417     // Pass the matrices to the shader
418     glUniformMatrix4fv(viewLoc, 1, GL_FALSE, glm::value_ptr(view));
419     glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr ➤
        (projection));
420
421     // Obtener matriz de Vista
422     view = camera.GetViewMatrix();
423
424     // Operar y dibujar modelo de PISO
425     glm::mat4 model(1);
426     model = glm::mat4(1);
427     model = glm::translate(model, glm::vec3(0.0f, 0.8f, 0.0f));
428     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
429     glUniform1i(glGetUniformLocation(lightningShader.Program, ➤
        "activaTransparencia"), 0);
430     Piso.Draw(lightningShader);
431
432     // Operar y dibujar modelo de HIDRANTE
433     model = glm::mat4(1);
434     model = glm::translate(model, glm::vec3(0.0f, 0.0f, 5.0f));
435     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
436     Hidrante.Draw(lightningShader);
437
438     // Operar y dibujar modelo de HABITACION GIMNASIO
439     model = glm::mat4(1);
440     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
441     Habitacion.Draw(lightningShader);
442
443     // Operar y dibujar modelos (ELEMENTO) Bancas inclinadas

```

```
444     model = glm::mat4(1);
445     model = glm::translate(model, glm::vec3(13.0f, 0.0f, -41.0f));
446     model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f,  ➤
        -1.0f, 0.0f));
447     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
448     Banca_inclinada.Draw(lampShader);
449
450     model = glm::mat4(1);
451     model = glm::translate(model, glm::vec3(13.0f, 0.0f, -33.0f));
452     model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f,  ➤
        -1.0f, 0.0f));
453     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
454     Banca_inclinada.Draw(lampShader);
455
456     // Operar y dibujar modelos (ELEMENTO) Caminadoras
457     model = glm::mat4(1);
458     model = glm::translate(model, glm::vec3(22.0f, 0.0f, -68.0f));
459     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
460     Caminadora.Draw(lampShader);
461
462     model = glm::mat4(1);
463     model = glm::translate(model, glm::vec3(13.0f, 0.0f, -68.0f));
464     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
465     Caminadora.Draw(lampShader);
466
467     model = glm::mat4(1);
468     model = glm::translate(model, glm::vec3(4.0f, 0.0f, -68.0f));
469     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
470     Caminadora.Draw(lampShader);
471
472     model = glm::mat4(1);
473     model = glm::translate(model, glm::vec3(-5.0f, 0.0f, -68.0f));
474     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
475     Caminadora.Draw(lampShader);
476
477     // Operar y dibujar modelos (ELEMENTO) Barras
478     model = glm::mat4(1);
479     model = glm::translate(model, glm::vec3(-20.0f, 0.0f, -64.0f));
480     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
481     Barra.Draw(lampShader);
482
483     model = glm::mat4(1);
484     model = glm::translate(model, glm::vec3(-28.0f, 0.0f, -53.0f));
485     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
486     Barra.Draw(lampShader);
487
488     // Operar y dibujar modelos (ELEMENTO) Racks mancuernas
489     model = glm::mat4(1);
490     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
```

```
491 Rack.Draw(lampShader);
492
493 model = glm::mat4(1);
494 glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
495 Rack2.Draw(lampShader);
496
497 // Operar y dibujar modelos (ELEMENTO) Mancuernas
498 model = glm::mat4(1);
499 model = glm::translate(model, glm::vec3(3.0f, 0.3f, 0.0f));
500 glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
501 Mancuerna.Draw(lampShader);
502 model = glm::mat4(1);
503 model = glm::translate(model, glm::vec3(-4.0f, 0.3f, 0.0f));
504 glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
505 Mancuerna.Draw(lampShader);
506 model = glm::mat4(1);
507 model = glm::translate(model, glm::vec3(-3.0f, 0.3f, 0.5f));
508 model = glm::rotate(model, glm::radians(15.0f), glm::vec3(0.0f, ➤
    -1.0f, 0.0f));
509 glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
510 Mancuerna.Draw(lampShader);
511
512 // Operar y dibujar modelo Estante
513 model = glm::mat4(1);
514 glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
515 Estante.Draw(lightningShader);
516
517 // Operar y dibujar modelo (ELEMENTO) Multiejercicios - Fondos
518 model = glm::mat4(1);
519 model = glm::translate(model, glm::vec3(-12.0f, 9.0f, -88.5f));
520 glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
521 Multi.Draw(lightningShader);
522
523 // Operar y dibujar modelos (ELEMENTO) Pesas Rusas
524 model = glm::mat4(1);
525 model = glm::translate(model, glm::vec3(-21.0f, 0.0f, -42.0f));
526 glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
527 Rusa1.Draw(lightningShader);
528 model = glm::mat4(1);
529 model = glm::translate(model, glm::vec3(-23.0f, 0.0f, -51.0f));
530 model = glm::rotate(model, glm::radians(30.0f), glm::vec3(0.0f, ➤
    1.0f, 0.0f));
531 glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
532 Rusa1.Draw(lightningShader);
533
534 model = glm::mat4(1);
535 model = glm::translate(model, glm::vec3(-21.0f, 0.0f, -52.0f));
536 glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
537 Rusa2.Draw(lightningShader);
```

```
538     model = glm::mat4(1);
539     model = glm::translate(model, glm::vec3(-20.2f, 0.0f, -55.0f));
540     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
541     Rusa2.Draw(lightningShader);
542
543     model = glm::mat4(1);
544     model = glm::translate(model, glm::vec3(-21.0f, 0.0f, -62.0f));
545     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
546     Rusa3.Draw(lightningShader);
547
548     // Operar y dibujar modelos (ELEMENTO) Pelotas Suizas
549     model = glm::mat4(1);
550     model = glm::translate(model, glm::vec3(-15.0f, 0.0f, -26.5f));
551     model = glm::scale(model, glm::vec3(1.5f, 1.5f, 1.5f));
552     model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
553     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
554     Suiza.Draw(lightningShader);
555
556     model = glm::mat4(1);
557     model = glm::translate(model, glm::vec3(-19.0f, 0.0f, -25.2f));
558     model = glm::scale(model, glm::vec3(1.5f, 1.5f, 1.5f));
559     model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
560     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
561     Suiza.Draw(lightningShader);
562
563     // Operar y dibujar modelos de Marcos Interiores
564     model = glm::mat4(1);
565     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
566     Marcos.Draw(lightningShader);
567
568     // Operar y dibujar modelo de Entrada
569     model = glm::mat4(1);
570     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
571     Entrada.Draw(lightningShader);
572
573     // Operar y dibujar modelo de Entrada - Puerta
574     model = glm::mat4(1);
575     model = glm::translate(model, glm::vec3(12.3f, 1.4f, -31.7f));
576     model = glm::rotate(model, glm::radians( rotDoor ), glm::vec3(0.0f, 1.0f, 0.0f));
577     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
578     Puerta.Draw(lightningShader);
579
580     // Operar y dibujar modelos (ELEMENTO) Accesorios - Gorras
581     model = glm::mat4(1);
582     model = glm::translate(model, glm::vec3(0.0f, 4.12f, 3.4f));
583     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
```

```
584      Gorra1.Draw(lightningShader);
585
586      model = glm::mat4(1);
587      model = glm::translate(model, glm::vec3(0.0f, 4.12f, 3.5f));
588      glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
589      Gorra2.Draw(lightningShader);
590
591      model = glm::mat4(1);
592      model = glm::translate(model, glm::vec3(0.0f, 4.12f, 3.7f));
593      glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
594      Gorra3.Draw(lightningShader);
595
596      model = glm::mat4(1);
597      model = glm::translate(model, glm::vec3(9.0f, 6.0f, 13.5f));
598      glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
599      Gorra3.Draw(lightningShader);
600
601      // Operar y dibujar modelo de Carro
602      model = glm::mat4(1);
603      model = glm::translate(model, PosIniCar + glm::vec3(movKitX, 0,  ➤
        movKitZ));
604      model = glm::rotate(model, glm::radians(rotKit), glm::vec3(0.0f,  ➤
        1.0f, 0.0));
605      model = glm::scale(model, glm::vec3(1.5f, 1.5f, 1.5f));
606      glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
607      Carro.Draw(lightningShader);
608
609      // Operar y dibujar modelo de Soportes de Camaras
610      model = glm::mat4(1);
611      model = glm::translate(model, glm::vec3(-12.0f, 15.0f, -82.6f));
612      model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));
613      glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
614      Soporte.Draw(lightningShader);
615
616      model = glm::mat4(1);
617      model = glm::translate(model, glm::vec3(-8.0f, 16.0f, -13.0f));
618      model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));
619      glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
620      Soporte.Draw(lightningShader);
621
622      // Operar y dibujar modelo de Camaras
623      model = glm::mat4(1);
624      model = glm::translate(model, glm::vec3(-12.0f, 15.0f, -82.6f));
625      model = glm::rotate(model, glm::radians(rotCam), glm::vec3(0.0f,  ➤
        1.0f, 0.0));
626      model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));
627      glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
628      Camara.Draw(lightningShader);
629
```

```
630     model = glm::mat4(1);
631     model = glm::translate(model, glm::vec3(-8.0f, 16.0f, -13.0f));
632     model = glm::rotate(model, glm::radians(rotCam), glm::vec3(0.0f, 1.0f, 0.0f));
633     model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));
634     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
635     Camara.Draw(lightingShader);
636
637     /* ----- Modelos con transparencia ----- */
638     glEnable(GL_BLEND); // Activa la funcionalidad para trabajar el
        canal alfa
639     glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
640
641     // Operar y dibujar modelo de Ventanas
642     model = glm::mat4(1);
643     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
644     glUniform1i(glGetUniformLocation(lightingShader.Program,
        "activaTransparencia"), 1);
645     glUniform4f(glGetUniformLocation(lightingShader.Program,
        "colorAlpha"), 0.0f, 0.0f, 0.0f, 0.05f);
646     Ventanas.Draw(lightingShader);
647
648     glDisable(GL_BLEND); //Desactiva el canal alfa
649     glUniform4f(glGetUniformLocation(lightingShader.Program,
        "colorAlpha"), 1.0f, 1.0f, 1.0f, 1.0f);
650     glBindVertexArray(0);
651
652
653     // Also draw the lamp object, again binding the appropriate shader
654     //lampShader.Use();
655     //// Get location objects for the matrices on the lamp shader
        (these could be different on a different shader)
656     //modelLoc = glGetUniformLocation(lampShader.Program, "model");
657     //viewLoc = glGetUniformLocation(lampShader.Program, "view");
658     //projLoc = glGetUniformLocation(lampShader.Program,
        "projection");
659
660     //// Set matrices
661     //glUniformMatrix4fv(viewLoc, 1, GL_FALSE, glm::value_ptr(view));
662     //glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr
        (projection));
663     //model = glm::mat4(1);
664     //model = glm::translate(model, lightPos);
665     //model = glm::scale(model, glm::vec3(0.2f)); // Make it a smaller
        cube
666     //glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr
        (model));
667     //// Draw the light object (using light's vertex attributes)
668     ////for (GLuint i = 0; i < 1; i++)
```



```

669         /////{
670         /////     model = glm::mat4(1);
671         /////     model = glm::translate(model, pointLightPositions[i]);
672         /////     model = glm::scale(model, glm::vec3(0.2f)); // Make it a
        smaller cube
673         /////     glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr
        (model));
674         /////     glBindVertexArray(VAO);
675         /////     glDrawArrays(GL_TRIANGLES, 0, 36);
676         /////}
677
678         //glBindVertexArray(0);
679
680
681         /* ----- Animacion Shader -----*/
682         /*----- Personaje Animado (Abdominales) -----*/
683         animacionPersonaje.initShaders(animShader.Program);
684         animShader.Use();
685         modelLoc = glGetUniformLocation(animShader.Program, "model");
686         viewLoc = glGetUniformLocation(animShader.Program, "view");
687         projLoc = glGetUniformLocation(animShader.Program, "projection");
688
689         glUniformMatrix4fv(viewLoc, 1, GL_FALSE, glm::value_ptr(view));
690         glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr
        (projection));
691
692         glUniform3f(glGetUniformLocation(animShader.Program,
        "material.specular"), 0.5f, 0.5f, 0.5f);
693         glUniform1f(glGetUniformLocation(animShader.Program,
        "material.shininess"), 12.0f);
694         glUniform3f(glGetUniformLocation(animShader.Program,
        "light.ambient"), 0.75f, 0.75f, 0.75f);
695         glUniform3f(glGetUniformLocation(animShader.Program,
        "light.diffuse"), 0.75f, 0.75f, 0.75f);
696         glUniform3f(glGetUniformLocation(animShader.Program,
        "light.specular"), 0.5f, 0.5f, 0.5f);
697         glUniform3f(glGetUniformLocation(animShader.Program,
        "light.direction"), 0.0f, -1.0f, -1.0f);
698         view = camera.GetViewMatrix();
699
700         model = glm::mat4(1);
701         model = glm::translate(model, glm::vec3(PosIniPerson.x,
        PosIniPerson.y, PosIniPerson.z));
702         model = glm::scale(model, glm::vec3(0.06f)); // ESCALAR ANIMACION
        al 6%
703         glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
704         animacionPersonaje.Draw(animShader);
705         glBindVertexArray(0);
706

```



```

707
708
709     /* ----- SKYBOX Shader ----- */
710     // Atributos SKYBOX
711     glDepthFunc(GL_EQUAL); // Change depth function so depth test      ➤
712     passes when values are equal to depth buffer's content
713     SkyBoxshader.Use();
714     view = glm::mat4(glm::mat3(camera.GetViewMatrix())); // Remove    ➤
715     any translation component of the view matrix
716     glUniformMatrix4fv(glGetUniformLocation(SkyBoxshader.Program,    ➤
717         "view"), 1, GL_FALSE, glm::value_ptr(view));
718     glUniformMatrix4fv(glGetUniformLocation(SkyBoxshader.Program,    ➤
719         "projection"), 1, GL_FALSE, glm::value_ptr(projection));
720
721     // Dibujar SKYBOX
722     glBindVertexArray(skyboxVAO);
723     glActiveTexture(GL_TEXTURE1);
724     glBindTexture(GL_TEXTURE_CUBE_MAP, cubemapTexture);
725     glDrawArrays(GL_TRIANGLES, 0, 36);
726     glBindVertexArray(0);
727     glDepthFunc(GL_LESS); // Set depth function back to default
728
729     // Swap the screen buffers
730     glfwSwapBuffers(window);
731 }
732
733 // Terminate GLFW, clearing any resources allocated by GLFW.
734 glfwTerminate();
735 return 0;
736 }
737
738 /**
739  * \fn void DoMovement()
740  * \brief Modifica posiciones de Camara respecto a Entradas de Usuario
741  */
742 void DoMovement()
743 {
744     // Controles de Camara
745     if (keys[GLFW_KEY_W] || keys[GLFW_KEY_UP])
746     {
747         camera.ProcessKeyboard(FORWARD, deltaTime);
748     }
749     if (keys[GLFW_KEY_S] || keys[GLFW_KEY_DOWN])
750     {
751         camera.ProcessKeyboard(BACKWARD, deltaTime);

```

```
752     }
753
754     if (keys[GLFW_KEY_A] || keys[GLFW_KEY_LEFT])
755     {
756         camera.ProcessKeyboard(LEFT, deltaTime);
757     }
758
759     if (keys[GLFW_KEY_D] || keys[GLFW_KEY_RIGHT])
760     {
761         camera.ProcessKeyboard(RIGHT, deltaTime);
762     }
763
764
765     // Control de Animacion Puerta
766     if (keys[GLFW_KEY_F])
767     {
768         actionDoor = true;
769     }
770
771     // Control de Animacion Coche
772     if (keys[GLFW_KEY_Z])
773     {
774         circuito = true;
775     }
776
777     if (keys[GLFW_KEY_X])
778     {
779         circuito = false;
780     }
781
782
783 }
784
785
786 /**
787  * \fn void animacion()
788  * \brief Realiza animaciones de objetos, modificando las variables para
789  * operaciones basicas
790  */
791 void animacion()
792 {
793     //Movimiento de Camara Seguridad
794     rotCam += (CamDerecha) ? 0.3f : -0.3f ;
795     CamDerecha = (rotCam >= 90.0f) ? false : CamDerecha;
796     CamDerecha = (rotCam <= -90.0f) ? true : CamDerecha;
797
798
799     //Movimiento de Puerta
```

```
800     if (actionDoor) {
801         rotDoor += (openDoor) ? -0.8f : 0.8f ;
802         if (rotDoor <= 0.0f) {
803             openDoor = false;
804             actionDoor = false;
805         }
806         if (rotDoor >= 90.0f) {
807             openDoor = true;
808             actionDoor = false;
809         }
810     }
811
812
813     //Movimiento del coche
814     if (circuito)
815     {
816         if (recorrido1)
817         {
818             rotKit = 0.0f;
819             movKitX -= 0.2f;
820             if ( movKitX < -35.0f )
821             {
822                 recorrido1 = false;
823                 recorrido2 = true;
824             }
825         }
826
827         if (recorrido2)
828         {
829             rotKit = -45.0f;
830             movKitX -= 0.1f;
831             movKitZ -= 0.1f;
832             if ( movKitX < -50.0f && movKitZ < -15.0f )
833             {
834                 recorrido2 = false;
835                 recorrido3 = true;
836             }
837         }
838
839         if (recorrido3)
840         {
841             rotKit = 0.0f;
842             movKitX -= 0.05f;
843             if ( movKitX < -90.0f )
844             {
845                 recorrido3 = false;
846                 recorrido4 = true;
847             }
848         }
849     }
```

```
849
850     if (recorrido4)
851     {
852         rotKit = 45.0f;
853         movKitX -= 0.1f;
854         movKitZ += 0.1f;
855         if ( movKitX < -105.0f && movKitZ > 0.0f )
856         {
857             recorrido4 = false;
858             recorrido5 = true;
859         }
860     }
861
862
863     if (recorrido5)
864     {
865         rotKit = 0.0f;
866         movKitX -= 0.2f;
867         if (movKitX < -150.0f)
868         {
869             recorrido5 = false;
870             recorrido6 = true;
871         }
872     }
873
874     if (recorrido6)
875     {
876         rotKit = 90.0f;
877         movKitZ += 0.2f;
878         if ( movKitZ > 14.0f )
879         {
880             recorrido6 = false;
881             recorrido7 = true;
882         }
883     }
884
885     if (recorrido7)
886     {
887         rotKit = 180.0f;
888         movKitX += 0.2f;
889         if ( movKitX > 0.0f )
890         {
891             recorrido7 = false;
892             recorrido8 = true;
893         }
894     }
895
896     if (recorrido8)
897     {
```

```
898         rotKit = -90.0f;
899         movKitZ -= 0.2f;
900         if (movKitZ < 0.0f)
901         {
902             recorrido8 = false;
903             movKitX = 0.0f;
904             movKitZ = 0.0f;
905             recorrido1 = true;
906         }
907     }
908 }
909 }
910
911 }
912 }
913
914 /**
915  * \fn void KeyCallback()
916  * \brief Opera cada que se presiona/libera una tecla a través de GLFW
917  */
918 void KeyCallback(GLFWwindow *window, int key, int scancode, int action,
919                 int mode)
920 {
921     if (GLFW_KEY_ESCAPE == key && GLFW_PRESS == action)
922     {
923         glfwSetWindowShouldClose(window, GL_TRUE);
924     }
925
926     if (key >= 0 && key < 1024)
927     {
928         if (action == GLFW_PRESS)
929         {
930             keys[key] = true;
931         }
932         else if (action == GLFW_RELEASE)
933         {
934             keys[key] = false;
935         }
936     }
937 }
938 }
939
940 /**
941  * \fn void MouseCallback()
942  * \brief Procesa los movimientos del Mouse sobre la Camara en Ventana
943  * Principal
944  */
```

```
945 void MouseCallback(GLFWwindow *window, double xPos, double yPos)
946 {
947     if (firstMouse)
948     {
949         lastX = xPos;
950         lastY = yPos;
951         firstMouse = false;
952     }
953
954     GLfloat xOffset = xPos - lastX;
955     GLfloat yOffset = lastY - yPos; // Reversed since y-coordinates go ↗
        from bottom to left
956
957     lastX = xPos;
958     lastY = yPos;
959
960     camera.ProcessMouseMovement(xOffset, yOffset);
961 }
```