	<b>Manual de Prácticas</b>
Secretaría/División: División de Ingeniería Eléctrica	Área/Departamento: Ingeniería en Computación


## Laboratorio de Computación Gráfica e Interacción Humano Computadora

### Texturizado

**N° de practica: 06**

<b>Nombre completo de los alumnos</b>		<b>Firma</b>
<b>Arroyo Llanes Miguel Alejandro</b>		
<b>N° de brigada: 2</b>	<b>Fecha de ejecución: 29/03/2025</b>	<b>Grupo: 03</b>
<b>Calificación:</b>	<b>Profesor: Ing. Jose Roque Roman Guadarrama</b>	

<b>Elaborado por:</b>	<b>Revisado por:</b>	<b>Autorizado por:</b>	<b>Vigente desde:</b>
-----------------------	----------------------	------------------------	-----------------------

	<b>Manual de Prácticas</b>
Secretaría/División: División de Ingeniería Eléctrica	Área/Departamento: Ingeniería en Computación

### Comentario:

Para esta práctica se creo un dado de 8 caras al cual se le añadió el texturizado de una imagen donde se le asigno el espacio respectivo a cada cara de los dados un numero (en función de la imagen de referencia).

También se cargo el modelo del auto anteriormente hecho (con las correcciones señaladas por el profesor), se le hizo modificaciones a su modelo en la aplicación 3ds Max para que el modelo tuviera textura.

### Bloque de código:

Importe auto

```

46      //Importar Partes del auto
47      Model BaseAuto;
48      Model CofreAuto;
49      Model RuedaDerFroAuto;
50      Model RuedaIzqFroAuto;
51      Model RuedaDerTraAuto;
52      Model RuedaIzqTraAuto;
53
54      //Dado

```

```

341
342      BaseAuto = Model();
343      BaseAuto.LoadModel("Models/BaseAuto.obj");
344      CofreAuto = Model();
345      CofreAuto.LoadModel("Models/CofreAuto.obj");
346      RuedaDerFroAuto = Model();
347      RuedaDerFroAuto.LoadModel("Models/RuedaDerFroAuto.obj");
348      RuedaIzqFroAuto = Model();
349      RuedaIzqFroAuto.LoadModel("Models/RuedaIzqFroAuto.obj");
350      RuedaDerTraAuto = Model();
351      RuedaDerTraAuto.LoadModel("Models/RuedaDerTraAuto.obj");
352      RuedaIzqTraAuto = Model();
353      RuedaIzqTraAuto.LoadModel("Models/RuedaIzqTraAuto.obj");
354
355

```



# Manual de Prácticas

Secretaría/División: División de Ingeniería Eléctrica

Área/Departamento: Ingeniería  
en Computación

Escritura dado de 8 caras:

[illegible]

Main:

```

P06-319218460.cpp:1
Material.cpp:1
Practic6:1 (Ámbito global)
main()

317 int main()
318 {
319     mainWindow = Window(1366, 768); // 1280, 1024 or 1024, 768
320     mainWindow.Initialise();
321
322     CreateObjects();
323     CreateDad08();
324     CreateShaders();
325
326     camera = Camera(glm::vec3(0.0f, 0.0f, 0.0f), glm::vec3(0.0f, 1.0f, 0.0f));
327
328     brickTexture = Texture("Textures/brick.png");
329     brickTexture.LoadTextureA();
330     dirtTexture = Texture("Textures/dirt.png");
331     dirtTexture.LoadTextureA();
332     plainTexture = Texture("Textures/plain.png");
333     plainTexture.LoadTextureA();
334     pisoTexture = Texture("Textures/piso.tga");
335     pisoTexture.LoadTextureA();
336     dad0Texture = Texture("Textures/Dad08Caras.png");
337     dad0Texture.LoadTextureA();
338     logofiTexture = Texture("Textures/escudo_fi_color.tga");
339     logofiTexture.LoadTextureA();
340
341     BaseAuto = Model();
342     BaseAuto.LoadModel("Models/BaseAuto.obj");
343     CofreAuto = Model();
344     CofreAuto.LoadModel("Models/CofreAuto.obj");
345     RuedaDerProAuto = Model();
346     RuedaDerProAuto.LoadModel("Models/RuedaDerProAuto.obj");
347
348     RuedaIzqTraAuto = Model();
349     RuedaIzqTraAuto.LoadModel("Models/RuedaIzqTraAuto.obj");
350
351     std::vector<std::string> skyboxFaces;
352     skyboxFaces.push_back("Textures/Skybox/cupertin-lake_rt.tga");
353     skyboxFaces.push_back("Textures/Skybox/cupertin-lake_lf.tga");
354     skyboxFaces.push_back("Textures/Skybox/cupertin-lake_dn.tga");
355     skyboxFaces.push_back("Textures/Skybox/cupertin-lake_up.tga");
356     skyboxFaces.push_back("Textures/Skybox/cupertin-lake_bk.tga");
357     skyboxFaces.push_back("Textures/Skybox/cupertin-lake_ft.tga");
358
359     skybox = Skybox(skyboxFaces);
360
361     GLuint uniformSpecIntensity = 0, uniformModel = 0, uniformView = 0, uniformE
362     uniformSpecIntensity = 0, uniformShininess = 0;
363     GLuint uniformColor = 0;
364     glm::mat4 projection = glm::perspective(45.0f, (GLfloat)mainWindow.getBuf
365
366     glm::mat4 model(1.0);
367     glm::mat4 modelaux(1.0);
368     glm::vec3 color = glm::vec3(1.0f, 1.0f, 1.0f);
369     ///Loop mientras no se cierra la ventana
370     while (!mainWindow.getShouldClose())
371     {
372         GLfloat now = glfwGetTime();
373         deltaTime = now - lastTime;
374         deltaTime += (now - lastTime) / limitFPS;
375         lastTime = now;
376
377         //Recibir eventos del usuario
378         glfwPollEvents();
379
380     }
381
382
383

```



Manual de Prácticas

Secretaría/División: División de Ingeniería Eléctrica

Área/Departamento: Ingeniería en Computación

P06-319218460.cpp:1

Material.cpp

Practica6

(Ámbito global)

main()

```
385 camera.mouseControl(mainWindow.getXChange(), mainWindow.getYChange()
386
387 // Clear the window
388 glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
389 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
390 skybox.DrawSkybox(camera.calculateViewMatrix(), projection);
391 shaderList[0].UseShader();
392 uniformModel = shaderList[0].GetModelLocation();
393 uniformProjection = shaderList[0].GetProjectionLocation();
394 uniformView = shaderList[0].GetViewLocation();
395 uniformColor = shaderList[0].getColorLocation();
396 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr
397 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camer
398 glUniform3f(uniformEyePosition, camera.getCameraPosition().x, cam
399
400 color = glm::vec3(1.0f, 1.0f, 1.0f); //color blanco, multiplica a
401
402 model = glm::mat4(1.0);
403 model = glm::translate(model, glm::vec3(0.0f, -2.0f, 0.0f));
404 model = glm::scale(model, glm::vec3(30.0f, 1.0f, 30.0f));
405 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(mode
406 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
407
408 pisoTexture.UseTexture();
409 meshList[2]->RenderMesh();
410
411 //Dado de OpenGL
412 //Ejercicio 1: Texturizar su cubo con la imagen dado_animales ya
413 model = glm::mat4(1.0);
414 model = glm::translate(model, glm::vec3(-1.5f, 4.5f, -2.0f));
415
```

88 % No se encontraron problemas. Línea: 456 Carácter: 1 TABULACIONES

416 glUniformMatrix4fv(uniformModel, 1, GL\_FALSE, glm::value\_ptr(model));
417 dadoTexture.UseTexture();
418 meshList[4]->RenderMesh();
419
420 //Ejercicio 2: Importar el cubo texturizado en el programa de modelado
421 //La imagen dado\_animales ya optimizada por ustedes
422 /\*
423 //Dado importado
424 model = glm::mat4(1.0);
425 model = glm::translate(model, glm::vec3(-3.0f, 3.0f, -2.0f));
426 model = glm::scale(model, glm::vec3(0.05f, 0.05f, 0.05f));
427 glUniformMatrix4fv(uniformModel, 1, GL\_FALSE, glm::value\_ptr(model));
428 Dado\_M.RenderModel();
429 \*/
430
431 //Reporte de práctica :
432 Ejercicio 1: Crear un dado de 8 caras y texturizarlo por medio de cód
433 Ejercicio 2: Importar el modelo de su coche con sus 4 llantas acomoda
434 y tener texturizadas las 4 llantas (diferenciar caucho y rin) y
435 texturizar el logo de la Facultad de ingeniería en el cofre de su pro
436
437 \*/
438
439 model = glm::mat4(1.0f);
440 float avanceZ = mainWindow.getavanzar();
441
442 // movimiento global del coche
443 model = glm::translate(model, glm::vec3(0.0f, 0.0f, avanceZ));
444 modelaux = model;
445
446 // Base del coche
447 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -1.5f));
448 model = glm::scale(model, glm::vec3(0.1f));
449 glUniformMatrix4fv(uniformModel, 1, GL\_FALSE, glm::value\_ptr(model));
450

88 % No se encontraron problemas. Línea: 1 Carácter: 1 TABULACIONES

P06-319218460.cpp:1

Material.cpp

Practica6

(Ámbito global)

main()


```
454 BaseAuto.RenderModel();
455
456 // Cofre
457 model = modelaux;
458 model = glm::translate(model, glm::vec3(0.1f, 1.95f, 0.8f));
459 model = glm::rotate(model, glm::radians(mainWindow.getarticulacio
460 model = glm::scale(model, glm::vec3(0.1f));
461 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(mode
462 glBindTexture(GL_TEXTURE_2D, 0);
463 CofreAuto.RenderModel();
464
465 // Rueda delantera derecha
466 model = modelaux;
467 model = glm::translate(model, glm::vec3(-2.0f, 0.0f, 2.1f));
468 model = glm::rotate(model, glm::radians(mainWindow.getarticulacio
469 model = glm::scale(model, glm::vec3(0.1f));
470 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(mode
471 RuedaDerFroAuto.RenderModel();
472
473 // Rueda delantera izquierda
474 model = modelaux;
475 model = glm::translate(model, glm::vec3(2.0f, 0.0f, 2.1f));
476 model = glm::rotate(model, glm::radians(mainWindow.getarticulacio
477 model = glm::scale(model, glm::vec3(0.1f));
478 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(mode
479 RuedaIzqFroAuto.RenderModel();
480
481 // Rueda trasera derecha
482 model = modelaux;
483
```

88 % No se encontraron problemas. Línea: 1 Carácter: 1 TABULACIONES

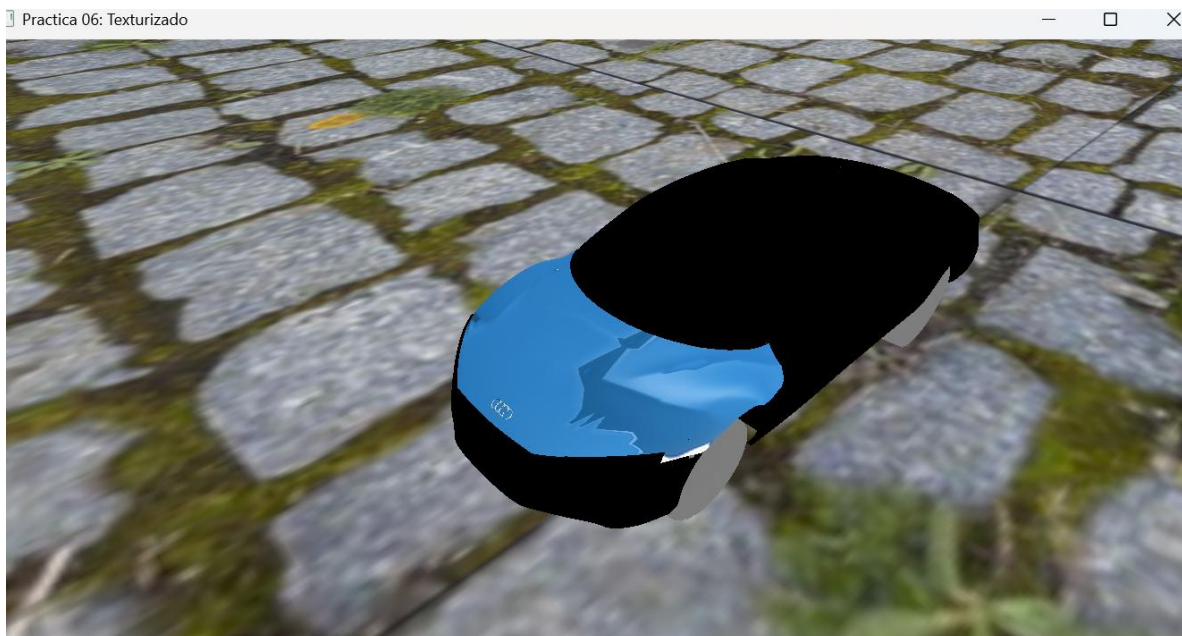
479 model = glm::scale(model, glm::vec3(0.1f));
480 glUniformMatrix4fv(uniformModel, 1, GL\_FALSE, glm::value\_ptr(model));
481 RuedaIzqFroAuto.RenderModel();
482
483 // Rueda trasera derecha
484 model = modelaux;
485 model = glm::translate(model, glm::vec3(-2.0f, 0.0f, -5.6f));
486 model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()
487 model = glm::scale(model, glm::vec3(0.1f));
488 glUniformMatrix4fv(uniformModel, 1, GL\_FALSE, glm::value\_ptr(model));
489 RuedaDerTraAuto.RenderModel();
490
491 // Rueda trasera izquierda
492 model = modelaux;
493 model = glm::translate(model, glm::vec3(2.0f, 0.0f, -5.6f));
494 model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()
495 model = glm::scale(model, glm::vec3(0.1f));
496 glUniformMatrix4fv(uniformModel, 1, GL\_FALSE, glm::value\_ptr(model));
497 RuedaIzqTraAuto.RenderModel();
498
499 glUseProgram(0);
500 mainWindow.swapBuffers();
501
502 return 0;
503
504
505
506
507 //blending: transparencia o traslucidez
508 glEnable(GL\_BLEND);
509 glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);
510 logoTexture.UseTexture(); //textura con transparencia o traslucidez
511 FIGURA A RENDERIZAR de OpenGL, si es modelo importado no se declara U
512 glDisable(GL\_BLEND);
513


88 % No se encontraron problemas. Línea: 1 Carácter: 1 TABULACIONES



	<b>Manual de Prácticas</b>
Secretaría/División: División de Ingeniería Eléctrica	Área/Departamento: Ingeniería en Computación

## Ejecución:



	<b>Manual de Prácticas</b>
Secretaría/División: División de Ingeniería Eléctrica	Área/Departamento: Ingeniería en Computación

### **Conclusión:**

Se han logrado aplicar y reforzar los conceptos previamente estudiados, consolidando el uso de jerarquías de transformación y texturización en modelos 3D importados

Durante el desarrollo de esta práctica también se avanzó en la comprensión del manejo de archivos .obj y .mtl, logrando una asignación automática de texturas y colores definidos desde el modelo original. A pesar de que el proceso no estuvo exento de errores, y por falta de tiempo no se logró de manera satisfactoria todo el texturizado, aunque reconozco que ya poseo suficiente conocimiento para lograrlo más adelante.

Si bien aún existen aspectos por pulir, como el dominio total del sistema de materiales o la optimización del uso de texturas, se reconoce un progreso real en el manejo de estas herramientas.

### **Bibliografía:**

1. No se buscó en ninguna fuente externa.