



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e  
INTERACCIÓN HUMANO COMPUTADORA



## **REPORTE DE PRÁCTICA N° 06**

**NOMBRE COMPLETO:** García Soto Jean Carlo

**N° de Cuenta:** 319226304

**GRUPO DE LABORATORIO:** 03

**GRUPO DE TEORÍA:** 04

**SEMESTRE 2025-2**

**FECHA DE ENTREGA LÍMITE:** 29/03/2025

**CALIFICACIÓN:** \_\_\_\_\_

## REPORTE DE PRÁCTICA:

1.- Ejecución de los ejercicios que se dejaron, comentar cada uno y capturas de pantalla de bloques de código generados y de ejecución del programa.

### Ejercicio 1: Crear un dado de 8 caras y texturizarlo por medio de código.

Para la creación del octaedro, se necesitó primero crear una función la cual nos creara esta figura, se usó como base el código del cubo, dándole 8 parámetros por cada vértice y 3 coordenadas por triángulo, modificamos de acuerdo con lo anterior la cantidad de parámetros a recibir y la cantidad de vértices. Y es importante importar la imagen para la textura, ya que esta se realizará a código.

```
38
39 Texture brickTexture;
40 Texture dirtTexture;
41 Texture plainTexture;
42 Texture pisoTexture;
43 Texture dadoTexture;
44 Texture logofitTexture;
45 Texture octaedroTexture;
...

262 void CrearOctaedro()
263 {
264     unsigned int octaedro_indices[] = {
265         // Parte superior
266         0, 1, 2, // Cara 1
267         3, 4, 5, // Cara 2
268         6, 7, 8, // Cara 3
269         9,10,11, // Cara 4
270         12,13,14, // Cara 5
271         15,16,17, // Cara 6
272         18,19,20, // Cara 7
273         21,22,23 // Cara 8
274     };
275 }
276
277 GLfloat octaedro_vertices[] = {
278     // Cara 1: top - left - front normal: (-1, 1, 1)
279     //x y z S T NX NY NZ
280     0.0f, 1.0f, 0.0f, 0.14f, 0.65f, -1.0f, 1.0f, 1.0f, //Punta
281     -1.0f, 0.0f, 0.0f, 0.02f, 0.34f, -1.0f, 1.0f, 1.0f, //Izquierda
282     0.0f, 0.0f, 1.0f, 0.26f, 0.34f, -1.0f, 1.0f, 1.0f, //Derecha
283
284     // Cara 2: top - front - right normal: (1, 1, 1)
285     //x y z S T NX NY NZ
286     0.0f, 1.0f, 0.0f, 0.285f, 0.35f, 1.0f, 1.0f, 1.0f,
287     0.0f, 0.0f, 1.0f, 0.4f, 0.65f, 1.0f, 1.0f, 1.0f,
288     1.0f, 0.0f, 0.0f, 0.16f, 0.65f, 1.0f, 1.0f, 1.0f,
289
290     // Cara 3: top - right - back normal: (1, 1, -1)
291     //x y z S T NX NY NZ
292     0.0f, 1.0f, 0.0f, 0.43f, 0.65f, 1.0f, 1.0f, -1.0f,
293     1.0f, 0.0f, 0.0f, 0.3f, 0.34f, 1.0f, 1.0f, -1.0f,
294     0.0f, 0.0f, -1.0f, 0.55f, 0.34f, 1.0f, 1.0f, -1.0f,
295
296     // Cara 4: top - back - left normal: (-1, 1, -1)
297     //x y z S T NX NY NZ
298     0.0f, 1.0f, 0.0f, 0.573f, 0.35f, -1.0f, 1.0f, -1.0f,
299     0.0f, 0.0f, -1.0f, 0.68f, 0.65f, -1.0f, 1.0f, -1.0f,
300     -1.0f, 0.0f, 0.0f, 0.45f, 0.65f, -1.0f, 1.0f, -1.0f,
301
302     // Cara 5: bottom - front - left normal: (-1, -1, 1)
303     //x y z S T NX NY NZ
304     0.0f, -1.0f, 0.0f, 0.715f, 0.64f, -1.0f, -1.0f, 1.0f,
305     0.0f, 0.0f, 1.0f, 0.6f, 0.35f, -1.0f, -1.0f, 1.0f,
306     -1.0f, 0.0f, 0.0f, 0.84f, 0.35f, -1.0f, -1.0f, 1.0f,
307
308     // Cara 6: bottom - right - front normal: (1, -1, 1)
309     //x y z S T NX NY NZ
310     0.0f, -1.0f, 0.0f, 0.86f, 0.35f, 1.0f, -1.0f, 1.0f,
311     1.0f, 0.0f, 0.0f, 0.98f, 0.64f, 1.0f, -1.0f, 1.0f,
312     0.0f, 0.0f, 1.0f, 0.75f, 0.64f, 1.0f, -1.0f, 1.0f,
313
314     // Cara 7: bottom - back - right normal: (1, -1, -1)
315     //x y z S T NX NY NZ
316     0.0f, -1.0f, 0.0f, 0.57f, 0.99f, 1.0f, -1.0f, -1.0f,
317     0.0f, 0.0f, -1.0f, 0.45f, 0.68f, 1.0f, -1.0f, -1.0f,
318     1.0f, 0.0f, 0.0f, 0.68f, 0.68f, 1.0f, -1.0f, -1.0f,
319 }
```

```

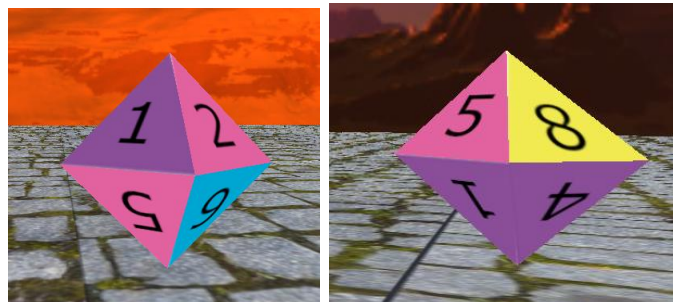
319 // Cara 8: bottom - left - back normal: (-1, -1, -1)
320 //x y z S T NX NY NZ
321 0.0f, -1.0f, 0.0f, 0.44f, 0.02f, -1.0f, -1.0f, -1.0f,
322 -1.0f, 0.0f, 0.0f, 0.55f, 0.3f, -1.0f, -1.0f, -1.0f,
323 0.0f, 0.0f, -1.0f, 0.31f, 0.3f, -1.0f, -1.0f, -1.0f
324 };
325
326 Mesh* octaedro = new Mesh();
327 octaedro->CreateMesh(octaedro_vertices, octaedro_indices, 192, 24);
328 meshList.push_back(octaedro);
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470 //Dado de OpenGL
471 //Ejercicio 1: Crear un dado de 8 caras y texturizarlo por medio de código
472 model = glm::mat4(1.0);
473 model = glm::translate(model, glm::vec3(0.0f, 5.0f, -5.0f));
474 model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f)); //Rotar con T
475 model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //Rotar con R
476 model = glm::rotate(model, glm::radians(mainWindow.getrota()), glm::vec3(1.0f, 0.0f, 0.0f)); //Rotar con E
477 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
478 octaedroTexture.UseTexture();
479 meshList[5]->RenderMesh();
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

Después con el software de GIMP se modificó la imagen que va a ser nuestra textura, se orientó a una forma en la que fuera fácil de manipular las coordenadas de los pixeles, y también pusimos los números en la figura, orientándolos de acuerdo en donde está la base de cada triángulo, para que a la hora de texturizarlo en OpenGL fuera más fácil manipularlo.



Se hizo que los números de la parte inferior de octaedro estuvieran de cabeza, cuidando que a la hora de mirarlos en la orientación correcta los números no estuvieran espejados.





**Ejercicio 2: Importar el modelo de su coche con sus 4 llantas acomodadas y tener texturizadas las 4 llantas (diferenciar caucho y rin)**

**Ejercicio 3: Texturizar la cara del personaje de la imagen tipo cars en el espejo (ojos) y detalles en cofre y parrilla de su propio modelo de coche**

En estos dos ejercicios se hizo el texturizado por medio de blender, dado que se reutilizó el modelo del carro de la practica pasada, abrimos los modelos .obj y texturizamos cada cosa que se nos pedía.

Ya en código solo era cosa de importar y ubicar nuestros modelos ya texturizados. Se le agregaron de nuevo las rotaciones que ya se tenían de la práctica pasada y al mismo tiempo se respetó la jerarquización.

```

50
51 //Modelos para ejercicio de la practica
52 Model KitCarro_M; //ojos y carro
53 Model LlantaIzqDelantera_M;
54 Model LlantaDerDelantera_M;
55 Model LlantaIzqTrasera_M;
56 Model LlantaDerTrasera_M;
57 Model Rin_M;
58 Model Cofre_M;
59 Model Parrilla_M;
60

```

```

368
369 KitCarro_M = Model();
370 KitCarro_M.LoadModel("Models/Carro_Ojos.obj");
371 LlantaIzqDelantera_M = Model();
372 LlantaIzqDelantera_M.LoadModel("Models/LlantaIzqDelantera.obj");
373 LlantaIzqTrasera_M = Model();
374 LlantaIzqTrasera_M.LoadModel("Models/LlantaIzqTrasera.obj");
375 LlantaDerDelantera_M = Model();
376 LlantaDerDelantera_M.LoadModel("Models/LlantaDerDelantera.obj");
377 LlantaDerTrasera_M = Model();
378 LlantaDerTrasera_M.LoadModel("Models/LlantaDerTrasera.obj");
379 Rin_M = Model();
380 Rin_M.LoadModel("Models/Rin.obj");
381 Cofre_M = Model();
382 Cofre_M.LoadModel("Models/Cofre.obj");
383 Parrilla_M = Model();
384 Parrilla_M.LoadModel("Models/Parrilla.obj");

```

```

470 //Dado de Opengl
471 //Ejercicio 1: Crear un dado de 8 caras y texturizarlo por medio de código
472 model = glm::mat4(1.0f);
473 model = glm::translate(model, glm::vec3(0.0f, 5.0f, -5.0f));
474 model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f)); //Rotar con T
475 model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //Rotar con R
476 model = glm::rotate(model, glm::radians(mainWindow.getrotax()), glm::vec3(1.0f, 0.0f, 0.0f)); //Rotar con E
477 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
478 octaedroTexture.UseTexture();
479 meshList[5]->RenderMesh();
480
481 //Ejercicio 2
482 //Carro sin llantas y cofre
483 model = glm::mat4(1.0f);
484 model = glm::translate(model, glm::vec3(0.0f, -1.7f, -5.0f));
485 model = glm::translate(model, glm::vec3(mainWindow.getTransladaCarro(), 0.0f, 0.0f)); // Mover(transladar) con 1 y 2
486 modelaux = model;
487 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
488 KitCarro_M.RenderModel(); //Muestra carro y ojos
489
490 //Llanta derecha delantera
491 model = modelaux;
492 model = glm::translate(model, glm::vec3(-3.1f, 0.4f, -1.65f));
493 model = glm::rotate(model, glm::radians(mainWindow.getrotallantas()), glm::vec3(0.0f, 0.0f, 1.0f)); //Rotar con Z y X
494 modelaux2 = model;
495 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
496 LlantaDerDelantera_M.RenderModel(); //Muestra Llanta
497
498 //Rin derecha delantera
499 model = modelaux2;
500 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
501 Rin_M.RenderModel(); //Muestra rin
502
503 //Llanta derecha trasera
504 model = modelaux;
505 model = glm::translate(model, glm::vec3(2.7f, 0.4f, -1.65f));
506 model = glm::rotate(model, glm::radians(mainWindow.getrotallantas()), glm::vec3(0.0f, 0.0f, 1.0f));
507 modelaux2 = model;
508 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
509 LlantaDerTrasera_M.RenderModel(); //Muestra Llanta
510
511 //Rin derecha delantera
512 model = modelaux2;
513 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
514 Rin_M.RenderModel(); //Muestra rin
515
516 //Llanta Izquierda trasera
517 model = modelaux;
518 model = glm::translate(model, glm::vec3(2.7f, 0.4f, 1.65f));
519 model = glm::rotate(model, glm::radians(mainWindow.getrotallantas()), glm::vec3(0.0f, 0.0f, 1.0f));
520 modelaux2 = model;
521 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
522 LlantaIzqTrasera_M.RenderModel(); //Muestra Llanta
523
524 //Rin Izquierda trasera
525 model = modelaux2;
526 model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
527 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
528 Rin_M.RenderModel(); //Muestra rin
529
530 //Llanta Izquierda Delantera
531 model = modelaux;
532 model = glm::translate(model, glm::vec3(-3.1f, 0.4f, 1.65f));
533 model = glm::rotate(model, glm::radians(mainWindow.getrotallantas()), glm::vec3(0.0f, 0.0f, 1.0f));
534 modelaux2 = model;
535 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
536 LlantaIzqDelantera_M.RenderModel(); //Muestra Llanta
537
538 //Rin Izquierda Delantera
539 model = modelaux2;
540 model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
541 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
542 Rin_M.RenderModel(); //Muestra rin
543
544 //Cofre
545 model = modelaux;
546 model = glm::translate(model, glm::vec3(-2.15f, 1.99f, 0.0f));
547 model = glm::rotate(model, glm::radians(mainWindow.getrotaCofre()), glm::vec3(0.0f, 0.0f, 1.0f)); //Rotar con V y B
548 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
549 Cofre_M.RenderModel(); //Muestra cofre
550

```

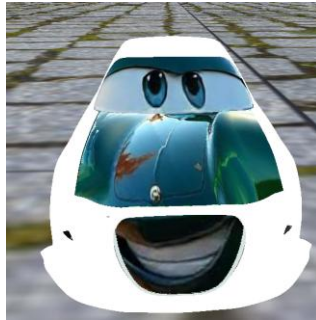


```

550
551 //Parrilla
552 model = modelaux;
553 model = glm::translate(model, glm::vec3(-4.82f, 0.95f, 0.0f));
554 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
555 Parrilla_M.RenderModel(); //Muestra parrilla
556
557

```

Dado que para poder visualizar un poco mejor la parrilla se imprimió o mostró desde otro modelo al igual que los rines, por lo que se tuvieron que hacer 2 modelos uno para el rin, y otro para la parrilla.



## **2.- Liste los problemas que tuvo a la hora de hacer estos ejercicios y si los resolvió explicar cómo fue, en caso de error adjuntar captura de pantalla**

En general no hubo problemas, solo que fue algo complicado a la hora de ubicar bien las normales de cada cara o triángulo del octaedro, ya que si se ponía en la posición equivocada la textura no se veía.

Y en el caso del carro, se me complicó a la hora de texturizar en blender, dado que hay muchos vértices y es complicado a la hora de ajustar los vértices en una estructura o en este caso la imagen donde las dimensiones no son las mismas dado que se enciman demasiado los vértices y se hay que tener que ubicar lo mejor posible para que el texturizado se lo más legible o limpio posible. Creo yo que fue el cofre y las llantas lo que más se me complicó.

## **3.- Conclusión:**

### **a. Los ejercicios del reporte: Complejidad, Explicación.**

Considero que los ejercicios fueron sencillos pero muy elaborados, dado que no teníamos previo manejo para texturizar, fue demasiado tardado, puede que haya mejores formas para poder realizarlo, por lo que siento que fue muy tardado a pesar de que el ejercicio era sencillo,

### **b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica**

Creo que estuvo bien explicado la práctica, lo único que creo que estaría bien es mencionar sobre las consideraciones que hay que tomar a la hora de escoger la imagen del carro para texturizarlo, ya que la imagen puede que no esté en el mejor ángulo para poder maniobrar o trabar para el texturizado, por lo que se llega a complicar bastante, y por ello pienso que estaría bien que esas consideraciones se puedan decir para escoger la imagen.

### **c. Conclusión**

Con esta práctica pude aprender el como texturizar por medio de código y en un software de modelado 3D por lo que ahora puedo entender el cómo se realiza este proceso, al igual de que consideraciones se puede hacer para facilitar texturizado.

## **1. Bibliografía en formato APA**

- G-Truc Creation. (s.f.). GLM 0.9.9 Manual. Recuperado de <https://chromium.googlesource.com/external/github.com/g-truc/glm/%2B0.9.9-a2/manual.md>
- Blender Foundation. (s.f.). *Selecting* — *Blender Manual*. Recuperado de <https://docs.blender.org/manual/en/2.80/modeling/meshes/selecting.html>